BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF RADIO ELECTRONICS
ÚSTAV RADIOELEKTRONIKY

RADIO MODULATION RECOGNITION NETWORKS
KLASIFIKACE RADIOVÝCH MODULACÍ POMOCÍ STROJOVÉHO UČENÍ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — **Kristýna Pijáčková**
AUTOR PRÁCE

SUPERVISOR — **doc. Ing. Tomáš Götthans, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Elektronika a komunikační technologie**

Ústav radioelektroniky

Studentka: Kristýna Pijáčková ***ID:*** 211534

Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Klasifikace radiových modulací pomocí strojového učení

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je navrhnout metody pro zpracování krátkých úseků signálů, které budou schopny spolehlivě identifikovat druh modulačního schématu. Při zpracování práce je kladen důraz na použití strojového učení, obzvláště pak neuronových sítí. Nejprve se seznamte s problematikou a možnostmi dostupných frameworků pro práci s neuronovými sítěmi, zaměřte se na práce s komplexními signály (v základním pásmu). Prostudujte si dostupné online knihovny signálů a proveďte trénink a evaluaci sítě. Případně vytvořte vlastní knihovnu signálů.

Vytvořte vlastní knihovnu reálných signálů pomocí radiového generátoru a softwarového rádia. Poté navrhněte skripty, které bude možno využít k tréninku a nasazení navržené neuronové sítě. Navrženou neuronovou síť ověřte na reálných signálech. Pokuste se optimalizovat velikost neuronové sítě a tím zajistit její možné nasazení na embedded platformě. Navrženou síť otestujte na minimální počet vzorků, které jsou potřebné k spolehlivé identifikaci.

DOPORUČENÁ LITERATURA:

[1] O'SHEA, Timothy J., Tamoghna ROY a T. Charles CLANCY. Over the Air Deep Learning Based Radio Signal Classification. Dostupné také z: https://arxiv.org/abs/1712.04578

[2] DEEPSIG INC.: Reinventing Wireless with Deep Learning [online]. Dostupné z: https://www.deepsig.io/

[3] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep Learning: (Adaptive Computation and Machine Learning series). The MIT Press, 2016. ISBN 978-0262035613.

[4] Dataset. Dostupné také z: https://www.deepsig.io/datasets

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 27.5.2021

Vedoucí práce: doc. Ing. Tomáš Götthans, Ph.D.

prof. Ing. Tomáš Kratochvíl, Ph.D.
předseda rady studijního programu

ABSTRACT

The bachelor thesis is focused on radio modulation classification with a deep learning approach. There are four deep learning architectures presented in the thesis. Three of them use convolutional and recurrent neural networks, and the fourth uses a transformer architecture. The final number of parameters of each model was considered during the design phase, as it can have a big impact on a memory footprint of a deployed model. The architectures were written in Keras, which is a software library, which provides a Python interface for neural networks. The results of the architectures were additionally compared to results from other research papers on this topic.

KEYWORDS

Radio modulation, classification, neural network, deep learning, Python, CNN, RNN

ABSTRAKT

Bakalářská práce se zabývá klasifikací rádiových modulací pomocí metod hloubkového učení. V práci jsou navrženy čtyři architektury, kde tři z nich jsou tvořeny pomocí konvolučních a rekurentních neuronových sítí a čtvrtá využívá architekturu transformátorů. Při návrhu architektur byl brán v potaz výsledný počet parametrů jednotlivých sítí, který může výrazně ovlivňovat výslednou velikost sítě. Pro účely návrhu byl využit programovací jazyk Python a knihovna Keras, která umožňuje práci s neuronovými sítěmi. Výsledky práce jsou následně zhodnoceny a porovnány s výsledky sítí navržených v článcích zabývajících se tímto tématem.

KLÍČOVÁ SLOVA

Rádiové modulace, klasifikace, neuronové sítě, hloubkové učení, Python, CNN, RNN

PIJÁČKOVÁ, Kristýna. *Radio Modulation Recognition Networks*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Radio Electronics, 2021, 61 p. Bachelor's Thesis. Advised by doc. Ing. Tomáš Götthans, Ph.D.

Rozšířený abstrakt

Bezdrátové technologie se postupně staly neodmyslitelnou součástí našeho života a jejich počet stále rapidně narůstá. Souvisle s tím však narůstají také požadavky na vlastnosti komunikačních systému, od kterých je požadovaná vyšší bitová rychlost, větší pokrytí sítí, nebo efektivnější využití spektra. Signál při přenosu navíc negativně ovlivňují interference, šumy a úniky, jejichž vlastnosti jsou při návrhu nejčastěji aproximované pomocí statistických modelů. Vzhledem k neustále narůstající složitosti při návrhu nových systémů, narůstá v posledních letech zájem o implementování metod strojového a hloubkového učení i v oblasti bezdrátové komunikace.

Tato práce je zaměřená na klasifikaci rádiových modulací pomocí hloubkového učení. Oproti aktuálně používaným metodám pro klasifikaci modulací, klasifikátor na základě hloubkového učení nepotřebuje žádné předchozí informace, nebo extrahované rysy z přijatého signálu. Klasifikátor může dále obsahovat větší počet modulací, aniž by se navýšila jeho složitost a má potenciál dosáhnout větší přesnosti i na krátkých úsecích signálu. Kromě určení typu modulace obdrženého signálu, která je potřebná pro následující demodulaci na straně přijímače, může být v kognitivním rádiu využitá např. i ke sledování spektra.

V rámci této práce byly navrženy čtyři modely, které jsou popsány v kapitole 4. Pro účely návrhu byl využit programovací jazyk Python a knihovna Keras, která umožňuje práci s neuronovými sítěmi. Při návrhu jejich architektur bylo dbáno na výslednou velikost sítě, která je zejména ovlivněna celkovým počtem parametrů. Tyto architektury jsou validovány na 4 datasetech a jejich výsledky jsou popsány v kapitole 6. Díky využití veřejně dostupných datasetů bylo možné dále porovnat výsledky z této práce s dostupnými výsledky z článků zabývajících se tímto tématem.

Nejlepších výsledků bylo v této práci dosaženo pomocí architektur CLDNN a CGDNN, jenž kombinovaly konvoluční a rekurentní sítě. Tyto architektury dosáhly velmi podobných výsledků a výsledná velikost jejich modelů byla 1,3Mb pro CLDNN a 636kB pro CGDNN. S ohledem na pozdější nasaditelnost na embedded platformy bych proto doporučovala využití CGDNN. Obě zmíněné architektury byly schopné dosáhnout lepších nebo alespoň porovnatelných výsledků s ohledem na ostatní články, ve kterých dané architektury obsahovaly až 40 krát více parametrů. Metoda klasifikace rádiových učení pomocí hloubkových sítí vykazuje velký potenciál, překážkou k využití této metody je však absence dostatečně diverzního datasetu naměřeného v reálném prostředí, který je důležitý pro vytvoření robustního klasifikátoru.

Author's Declaration

Author:	Kristýna Pijáčková
Author's ID:	211534
Paper type:	Bachelor's Thesis
Academic year:	2020/2021
Topic:	Radio Modulation Recognition Networks

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank my supervisor doc. Ing. Tomáš Götthans, Ph.D. for the opportunity to work on such an intriguing topic. I very much appreciate your time, encouragement, and support I received whilst working on this thesis.

Contents

List of Figures

List of Tables

Introduction

Wireless technology has rapidly developed over the years and became an inseparable part of our daily life. Each newly developed communication generation aims to achieve a wider coverage area, higher number of users, higher bit rate, improve the efficiency in spectrum usage and power consumption. The communication systems need to deal with diverse impairments, which negatively affect the transmitted signals. So far, the communication engineers used mainly statistical models, which would approximate the channel impairment effects, to design wireless systems. But the complexity of designing new communication systems is rapidly escalating with the highly increasing number of wireless devices [1]. Therefore, many researchers shift their focus to machine learning (ML) and deep learning (DL) techniques, especially with the beginning research in 6G.

ML or DL techniques can be used in wireless networks to improve radio resource allocation, energy efficiency, or dynamic spectrum access (DSA) [2]. Signal detection and modulation classification are essential for successful transmission of the information. Automatic modulation classification (AMC) can be also used in cognitive radio (CR), which was designed to scan its spectrum environment to enable more flexible DSA. By scanning the environment, it can adjust and modify parameters such as frequency, power, or bit-rate dynamically and improve the system's performance [3].

Current methods for AMC use either a feature-based approach or a likelihood-based approach. While the likelihood-based approach can be very accurate, it also demands a high computational complexity. The received signal has difficulties in finding a suitable analytical solution if there are many unknown parameters. Classifiers in the feature-based approach need diverse features to be extracted from the incoming signal. There is often a trade-off between the computational complexity, number of modulation schemes, and accuracy. In the past five years, there is a focus on another approach to AMC through deep learning. The DL approach does not require any pieces of information about the signal, nor does it need any additional features. The DL classifier can also include a wide range of modulation schemes, without any significant increase in its complexity.

This thesis is focused on radio modulation classification with deep learning architectures. There are four proposed architectures - CNN, CLDNN, CGDNN, and MCTransformer, which are trained and evaluated on four datasets. The main goal was to design architectures with a reduced number of parameters while maintaining high accuracy. The architectures proposed in this thesis have up to 50 times fewer parameters than other often-cited papers such as [4, 5]. They are evaluated on multiple datasets to get a better overview of their performance. The achieved results

are also compared with other research papers focused on this topic.

The first two chapters include an overview of a digital communication system and a brief description of the classification approaches. The third chapter is aimed to provide elementary knowledge to the reader on the topic of deep learning. It includes a brief introduction and description of three classes of deep neural networks used in the architectures' design in this thesis. The proposed architectures are listed in chapter four along with a layer visualization. This is followed by a description of the used radio modulation datasets in chapter five. Finally, chapter six presents results achieved in this thesis and their discussion.

1 Digital Communication Systems

Every communication system consists of three basic elements. Namely of a transmitter, a channel, and a receiver. When a source of information produces a signal containing a message, the transmitter will convert it to a suitable form matching the channel properties. The signal is then propagated over the channel to the receiver, which is located at a different place than the transmitter. However, due to the channel imperfections, the received signal is distorted. Fading and attenuation effects appear, and diverse noises are added to the transmitted signal. The receiver reconstructs the signal, so that an end-user gets a recognizable form of the original message. Fig. 1.1 shows a block diagram of the digital communication system.

This chapter aimed to provide an introduction to the communication systems since modulation classification an important part of it. The chapter includes brief descriptions of the communication system's blocks, more in-depth descriptions can be found in [6–8].

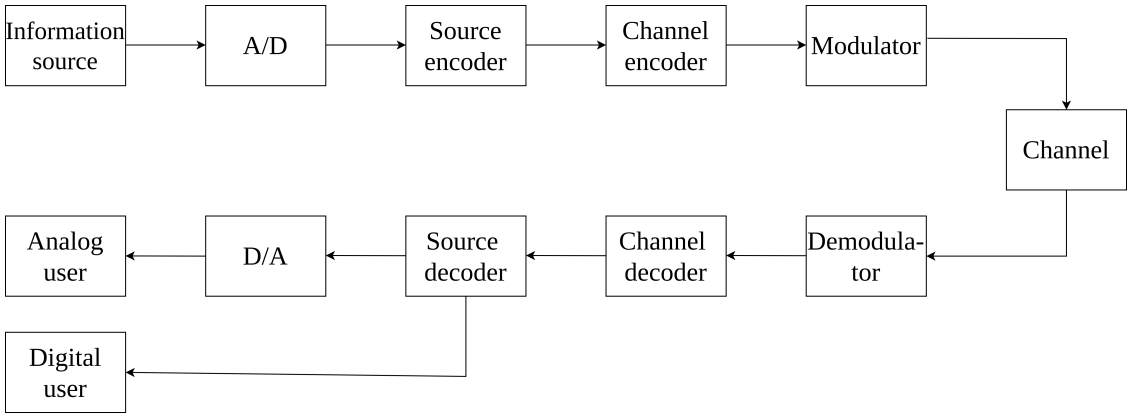


Fig. 1.1: Communication System [8]

1.1 Transmitter

This section describes operations in the transmitter, which are needed for successful signal transmission. These steps can be seen in the upper part of the Fig. 1.1.

1.1.1 Source of Information

If the input signal from the source is analog, an analog-to-digital converter is applied and converts the signal into a digital form.

In communication, a signal containing information is mostly a low-pass signal. It is a low-frequency signal with a spectrum located around zero frequency. Wireless communication channel, however, operates at higher frequencies not allowing the information signal to be directly transmitted over the channel. A translation to a band-pass signal with a higher frequency is therefore required to match the channel properties. For simplicity, the mathematical model introduced here is for a deterministic signal, which has no uncertainty about its time-dependent at any instant of time. However, many real-world signals require a probabilistic model, as they include many unknown factors.

The spectrum of a band-pass signal is located around the carrier frequency f_c. It is commonly used in radio communication. Compared with f_c, the bandwidth 2B of the signal is smaller and may be referred to as a narrowband signal. Any band-pass signal used in communication can be represented with a low-pass equivalent of the original signal. Working with low-pass equivalent makes it easier to handle the band-pass signal. The low-pass equivalent has lower rates of a sampled data due to lower required sampling rates.

The corresponding complex low-pass equivalent can be achieved by applying an analytic filter to the band-pass signal and shifting it to the origin $f = 0$ as is to be seen in Fig. 1.2.

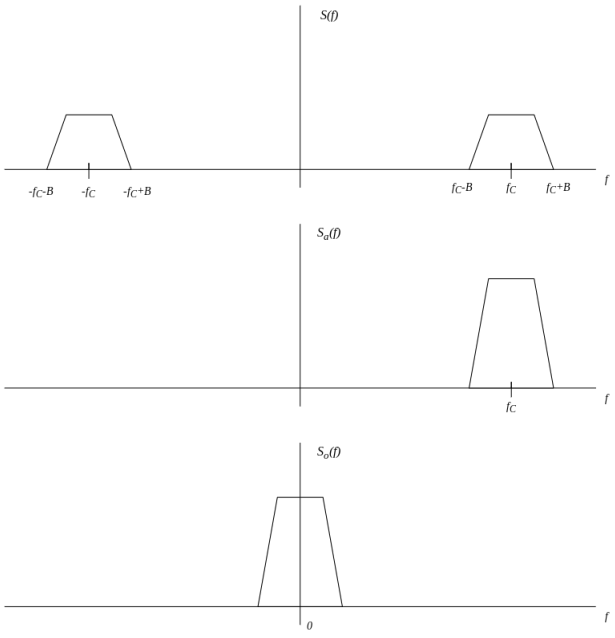


Fig. 1.2: Bandpass to analytic signal [9]

Hilbert Transform

Hilbert transform operates exclusively in a time domain and its usual application is on real-valued signals. It is a linear operation providing a $\pm 90°$ phase shift. It can be described with a Hilbert transform $H(f)$ which is defined as:

$$H(f) = -jsign(f) \tag{1.1}$$

where:

$$\begin{aligned} sign(f) &= -1 \quad for \quad f < 0 \\ &= \ \ 1 \quad for \quad f \geq 0. \end{aligned}$$

The spectrum of the transformed signal is then described as:

$$S_H(f) = H(f)S(f) = -jsign(f)S(f). \tag{1.2}$$

The magnitude spectrum of signal $s(t)$ and its Hilbert transform $s_H(t)$ are the same $|S(f)| = |S_H(f)|$. Phase response of Hilbert transform can be written as $arg[S(f)] = -arg[S_H(f)]$.

Analytic Signal

An analytic signal (or a pre-envelope) is a special type of complex signal with a single-sideband with a suppressed complex part of the signal. All negative frequency components can be eliminated in the real-valued signal $s(t)$ with a complex-valued signal called the pre-envelope, which can be described as:

$$s_a(t) = s(t) + js_H(t) \tag{1.3}$$

where $s_H(t)$ is the Hilbert transform of $s(t)$. Fourier transform of $s_a(t)$ may be written as:

$$S_a(f) = S(f) + sign(f)S(f) \tag{1.4}$$

then $S_a(f)$ can be rewritten as:

$$\begin{aligned} S_a(f) &= 2S(f) \quad for \quad f > 0 \\ &= S(0) \quad for \quad f = 0 \\ &= 0 \quad for \quad f < 0. \end{aligned}$$

The pre-envelope for negative frequencies can be described symmetrically.

Complex Envelopes of Band-Pass Signals

Complex envelope is represented by the signal:

$$s_e(t) = s_a(t)e^{-j2\pi f_c t} \tag{1.5}$$

where multiplying with the exponential $e^{-j2\pi f_c t}$ corresponds to a property of the Fourier transform and shifts the signal by f_c to its origin $f = 0$:

$$S_e(f) = S_a(f + f_c). \tag{1.6}$$

The original band-pass signal $s(t)$ can be expressed as:

$$s(t) = \mathbb{R}[s_e(t)e^{j2\pi f_c t}] \tag{1.7}$$

Cartesian Representation of Band-Pass Signals

Complex envelope $s_e(t)$ is a complex signal, which can be expressed as:

$$s_e(t) = s_I(t) + js_Q(t) \tag{1.8}$$

where $s_I(t)$ is referred to as in-phase component of the band-pass signal $s(t)$ and $s_Q(t)$ is referred to as quadrature-phase component of the signal $s_(t)$. Original band-pass signal $s(t)$ then responds to:

$$s(t) = s_I(t)\cos(2\pi f_c t) - s_Q(t)\sin(2\pi f_c t) \tag{1.9}$$

This equation shows how to reconstruct a band-pass signal from the complex envelope, this process is graphically pictured in Fig. 1.3.

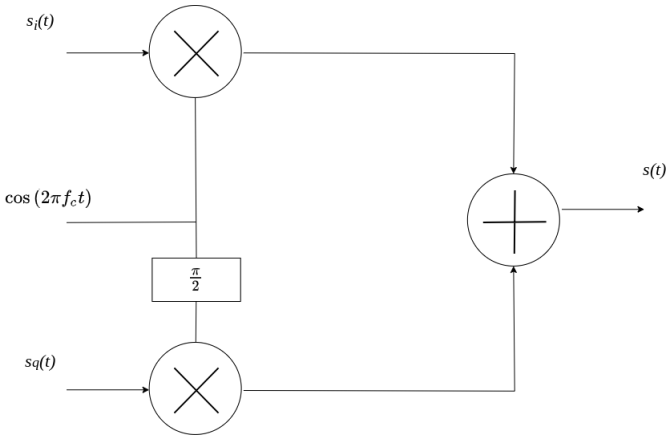


Fig. 1.3: Bandpass Signal from Complex Signal [9]

It is possible to obtain an equation for a reverse process and extract the complex envelope from the band-pass signal with an multiplication of the signal $s(t)$ with $\cos(2\pi f_c t)$:

$$2s(t)\cos(2\pi f_c t) = s_i(t) + s_i(t)\cos(4\pi f_c t) - s_q(t)\sin(4\pi f_c t) \tag{1.10}$$

and an multiplication of the signal $s(t)$ with $\sin(2\pi f_c t)$:

$$2s(t)\sin(2\pi f_c t) = -s_q(t) + s_i(t)\sin(4\pi f_c t) - s_q(t)\cos(4\pi f_c t) \tag{1.11}$$

with the use of following goniometric functions:

$$\begin{aligned} \cos\alpha\cos\beta &= \frac{1}{2}\cos(\alpha-\beta) + \frac{1}{2}\cos(\alpha+\beta) \\ \sin\alpha\cos\beta &= \frac{1}{2}\sin(\alpha-\beta) + \frac{1}{2}\sin(\alpha+\beta) \\ \sin\alpha\sin\beta &= \frac{1}{2}\cos(\alpha-\beta) - \frac{1}{2}\cos(\alpha+\beta). \end{aligned} \tag{1.12}$$

To suppress the modulated components at the frequency $4\pi f_c t$ and extract only the components of the complex envelope $s_i(t)$ and $s_q(t)$ a low-pass filter is applied as it is to be seen in 1.4.

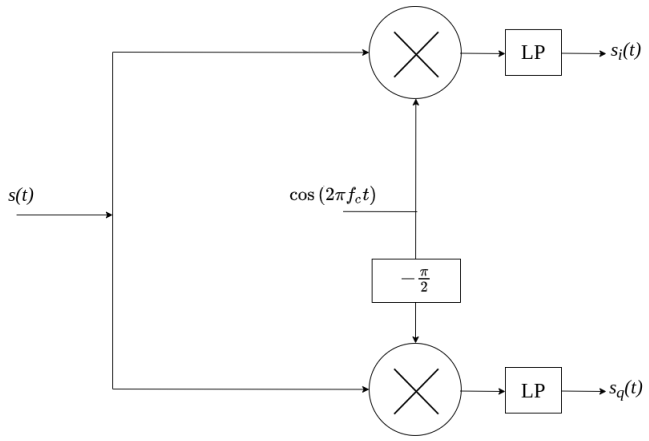


Fig. 1.4: Complex Envelope from Bandpass Signal [9]

Relationship Between Cartesian and Polar Representation

In polar representation the complex envelope is described as:

$$s_e(t) = a(t)e^{j\phi(t)} \tag{1.13}$$

where $a(t)$ is referred to as natural envelope of the band-pass signal $s(t)$ and $\phi(t)$ referred to as phase of the signal $s(t)$. Original band-pass signal $s(t)$ responds to:

$$s(t) = a(t)\cos\left[2\pi f_c t + \phi(t)\right]. \tag{1.14}$$

Relation between in-phase $s_I(t)$ and quadrature $s_Q(t)$ components and envelope $a(t)$ and phase $\phi(t)$ is described as:

$$a(t) = \sqrt{s_I^2(t) + s_Q^2(t)} \tag{1.15}$$

and

$$\phi(t) = \arctan\left(\frac{s_Q(t)}{s_I(t)}\right) \tag{1.16}$$

or conversely as:

$$s_I(t) = a(t)\cos\left[\phi(t)\right] \tag{1.17}$$

and

$$s_Q(t) = a(t)\sin\left[\phi(t)\right]. \tag{1.18}$$

Linear Band-Pass System

A linear system can be either described with its impulse response $h(t)$, which acts as a memory function of the system, or with its frequency response $H(f)$. A resulting response of the system $r(t)$ excited by the signal $s(t)$ is in time-domain given by convolution (symbolised as $*$):

$$r(t) = s(t) * h(t) = \int_{-\infty}^{+\infty} s(\tau)h(t-\tau)\,d\tau \tag{1.19}$$

where t stands for *response time*, τ stands for *excitation time* and $t - \tau$ is *system-memory time*.
In frequency-domain convolution corresponds to multiplying:

$$R(f) = S(f)H(f) \tag{1.20}$$

Except for the scaling factor 1/2, the complex envelope $r_e(t)$ of the output signal of a band-pass system is obtained by convolving the complex impulse response $h_e(t)$ of the system with the complex envelope $s_e(t)$) of the input band-pass signal, which plays a big role in computational terms [7].

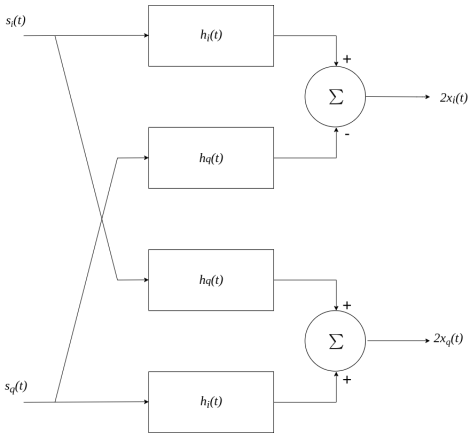


Fig. 1.5: Bandpass system [7]

Complex envelope of the band-pass system may be expressed as:

$$\begin{aligned} r_e(t) &= \frac{1}{2} s_e(t) * h_e(t) \\ 2r_e(t) &= [h_i(t) + jh_q(t)] * [s_i(t) + js_q(t)] \end{aligned} \tag{1.21}$$

in time-domain, where $h(t)$ is represented as $h(t) = \mathbb{R}[s_e(t)e^{j2\pi f_c t}]$ and $h_(t)$ is a complex envelope of impulse characteristic of the linear system. As convolution is distributive, the in-phase and quadrature components may be described as:

$$\begin{aligned} 2r_i(t) &= h_i(t) * s_i(t) - h_q(t) * s_q(t) \\ 2r_q(t) &= h_q(t) * s_i(t) + h_i(t) * s_q(t). \end{aligned} \tag{1.22}$$

In the frequency domain convolution is changed to multiplication:

$$R_e(f) = \frac{1}{2} S_e(f) H_e(f). \tag{1.23}$$

1.1.2 Source Encoder

The source encoder is responsible for eliminating redundant bits from the input information sequence. During the compressing, the number of bits is reduced and it can be classified either as lossless or lossy compression. For the lossless compression, the number of bits is reduced in a way, which allows perfect reconstruction of the source later on. This is not possible for the lossy compression and is subject to a maximum tolerable distortion. Either way, this block results in a reduced signal baud rate and a smaller bandwidth of the transmission channel.

1.1.3 Channel Encoder

The channel encoder adapts the transmitted signal to the transmission channel and, unlike the source encoder, adds redundancy to the transmitted signal. The added redundancy from the channel encoder helps with a correction of errors at the receiver. These errors occur during the transmission and can be caused by noises, signal leakage, various types of interference, etc...

1.1.4 Modulator

Modulation is a technique that changes the characteristics of the carrier frequency in accordance with the input signal [10]. The transmitted information can be either analog or digital and the carrier for both is a high-frequency sinusoidal signal. The modulation schemes might be therefore divided into analog and digital modulations also referred to as keying. A sinusoidal signal has three parameters: amplitude, frequency, and phase, which results in three basic modulation methods. Fig. 1.6 shows three basic binary modulations: amplitude shift keying, frequency-shift keying, and phase-shift keying. A variety of modulation schemes can be derived based on these three basic schemes.

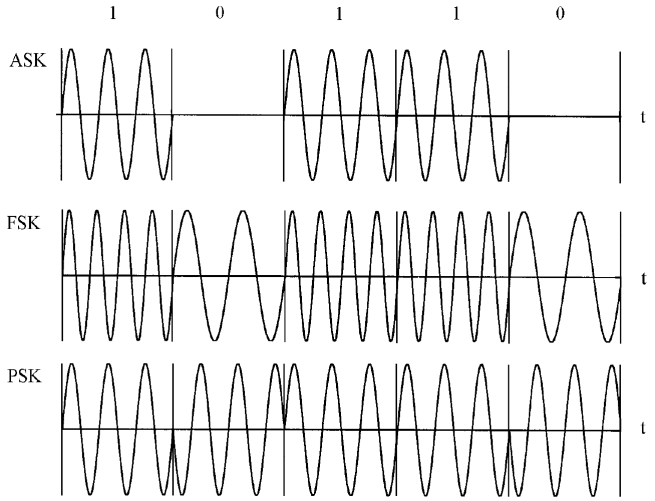


Fig. 1.6: Basic bandpass modulation schemes [8]

Some of the theoretical requirements on modulations schemes are either high data transmission rate, low transmission power, narrow signal bandwidth, robustness against interference, low computational power, or low error rate. The chosen modulation scheme should be therefore chosen carefully and it should consider the channel environment.

1.2 Wireless Communication Channel

For optimized performance and the right choice of modulation schemes, the knowledge of channel characteristics is important. Fig. 1.7 is a diagram block of a simplified digital communication model. The channel in this diagram consists of three elements. The first element is the channel filter with transfer function:

$$H(t) = H_T(t)H_C(t)H_R(t) \tag{1.24}$$

composed of the transfer function of the transmitter, the channel, and the receiver. The second element represents fading and the factor $A(t)$ is generally complex. Last element $n(t)$ is an additive noise and interference term.

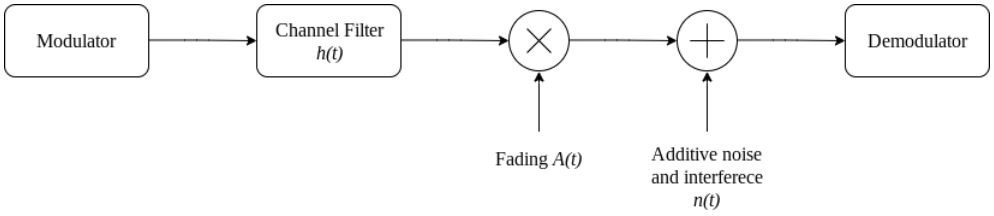


Fig. 1.7: Simplified block of a communication system for modulation and demodulation [8]

Received signal at the demodulator might be expressed as:

$$r(t) = A(t)[s(t) * h(t)] + n(t). \tag{1.25}$$

Generating different signals for modulation classification also uses the math models for better real-world approximation. Some of the popular channel models are therefore introduced in this section.

1.2.1 Additive White Gaussian Noise Channel

Additive white Gaussian noise (AWGN) is a widely used noise model and is considered to be a fundamental limitation to accurate modulation classification. It allows modeling channels with predominant thermal noise. The amplitude-frequency response of the channel is flat with unlimited bandwidth and its phase frequency response is linear for all frequencies. The channel does not cause fading and the channel only adds the AWGN $(n(t))$to the passing signal $s(t)$:

$$r(t) = s(t) + n(t) \tag{1.26}$$

1.2.2 Fading Channel

Fading can be generally categorized as slow and fast. Slow fading occurs when the transmitted signal is obscured by a large object and can be referred to as shadowing. The cause of fast fading is interference between multiple versions of the transmitted signal. Diffraction, absorption, or reflection of the original signal creates attenuated copies and each copy arrives at the receiver at a slightly different time. Different Doppler shift caused by moving receiver and transmitter results in carrier frequency shift and spreading of signal bandwidth.

The properties of the transmitted signal are changed by fading. Amplitude fluctuation and phase variation or intersymbol interference are other causes of the fading. In modulation classification, it is required that the classifier is robust in fading channels.

1.2.3 Non-Gaussian Noise Channel

Non-Gaussian noise channels are complex to model as the noises are impulsive. The impulses noises have a higher probability for high power noise components, unlike Gaussian noise. They are caused by man-made sources in form of incidental electromagnetic radiation. Modeling those noises is complex, but helps to approximate the signal to real-world scenarios.

1.3 Receiver

In the receiver, the transmitted signal undergoes a similar process as in the transmitter, but in reverse order. The signal needs to be demodulated first, after that it is passed to the channel decoder and source decoder. If the signal should be received as digital, it can be passed to the user. To obtain an analog signal, it needs to be additionaly passed through a digital-to-analog converter.

It is unrealistic to assume, that the channel was affected only with AWGN and is perfectly synchronized between transmitter and receiver. There are often distortions, phase, and frequency shifts and delays included in the received signal. To achieve optimal detection, despite an inter-symbol interference caused by the distortion and asynchronicity, the receiver must correct these issues. Because of this, the receiver also includes a synchronization, which can be either blind or data-aided, and equalization.

2 Modulation Classification

Having an accurate classifier as a part of a communication system is crucial. With incorrect classification, the demodulator will not be able to use a correct demodulation method and the entire transmission fails. The numerous conditions affecting the channel such as fading, Doppler shift, AWGN and many others are challenging for the classifier. It is necessary to consider them when designing the classifier for its practical applications. The knowledge of many modulation types is important as well and it should operate with limited knowledge of the channel scenarios. Another thing to consider is the complexity as it affects the hardware choices as well as the processing time of the computation. Therefore the key to completing the transmission of the signal and recovering the captured message is a robust, versatile, computationally efficient, and accurate classifier.

There are currently two approaches used for classifying radio modulations, namely likelihood-based (LB) and feature-based (FB) approach. This thesis focuses on a new possible approach for classification, which uses deep neural networks. This chapter provides an introduction to those three methods, more information can be found in [11–13].

2.1 Likelihood-Based Approach

The likelihood-based approach is based on multiple composite hypothesis-testing problems and consists of two steps. In the first one, a likelihood is evaluated for each modulation hypothesis with observed signal samples. The derived likelihood functions can be further modified to obtain lower computational complexity or to be applicable in non-cooperative environments. In the second step, a conclusion of the classification decision is made by comparing the likelihood functions of different modulation hypotheses [11].

This method provides optimal performance for the classification. But as the number of unknown parameters increases, obtaining an exact analytic solution for the decision function becomes difficult. There is also a trade-off between the classifier's complexity and its performance [12].

2.2 Feature-Based Approach

The feature-based approach consists of two main parts, namely a feature-extraction and a classifier. There are no rules for selecting the right features, but they should be sensitive to the modulations and insensitive to noisy channels and varying SNR.

Spectral, statistical, spectrum and constellation shape features are the most common features mentioned in the literature. Each feature comes with its advantages and disadvantages. Spectral features have low complexity but are sensitive to additive noises. High-order statistics are resistant to AWGN and multipath channels and are sensitive in discriminating between modulation schemes such as M-PSK and M-QAM. Cyclostationary features are computationally complex, but have good resistance at low SNR, etc... [13]

After the features are extracted, they are passed to a classifier, which often uses machine learning. Some of the common classifier approaches are decision trees, artificial neural networks, support vector machines, or k-nearest neighbors.

The FB approach might be easier to implement than the LB approach. Its complexity and performance depend mainly on the chosen features which are to be extracted from the signals. Better accuracy and higher robustness very often come at a price of higher computational complexity.

2.3 Deep Learning Approach

Using deep learning for modulation classification is a recent approach to the problem, which gained focus over the last five years. What is different about this approach is that the input signal can be directly passed to a DL model for classification. It can work without any feature extraction, and it does not need to know anything about the parameters of the signal. The classifier can include a huge number of modulations without any significant increase in computational complexity.

When it comes to deep learning architectures, there are innumerous possibilities for different tasks. Choosing or designing the right architecture might be challenging as the performance depends on many variables. Hyper-parameters are variables that determine the network structure, such as a number of hidden layers and units, activation functions, optimizer with correct learning rates, etc... and they can affect the networks performance and size a lot. However, thanks to transfer learning, it might become quite user-friendly and easily re-usable, once there are few available models. Transfer learning is a method, where weights from a pre-trained model trained on similar data are taken and are used as a learning starter point and thus allows faster training with higher accuracy on a smaller dataset than it would if one would train a model from scratch.

The most commonly used architectures, which are to be seen in research papers such as [4, 5, 14–17], are either convolutional or recurrent networks or their combination, which are described in the next chapter.

3 Deep Learning

Deep learning (DL) has reached huge popularity over the past 10 years. It became state-of-the-art in computer vision, as well as natural language processing and it finds its way to other fields as well. As the practical part of this thesis is based on DL architectures, this chapter includes a brief introduction to DL. For anyone interested more in the theory and the principles, there are chapters dedicated to the mentioned architectures in [18] or [19]. They offer more detailed description, including the math behind it, and [19] includes interactive code examples as well.

3.1 Introduction

DL is a mathematical framework for learning representations from data [20]. The data representations are obtained by layered neural networks (in most cases), which are referred to as layers.

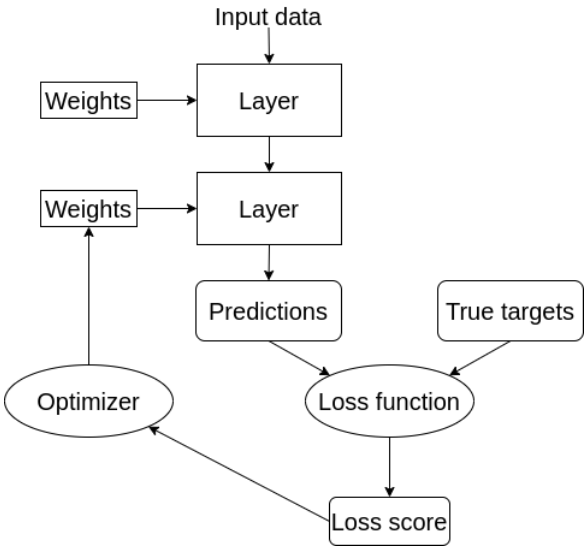


Fig. 3.1: Working principle of DL [20]

Input-to-target mapping tasks are done with a deep sequence of data transformations learned by exposure to training data. This process can be seen in Fig. 3.1. To begin with, the data are passed onto the layers inside the DL model. The rule of thumb in machine learning is to split provided dataset into an 80:20 ratio of training and testing data. We can further split the data and create a third validation set, which helps with an observation of the model performance during the training. The layer tries to find useful features in the input data and stores them

in form of weights. The weights (also called parameters of a layer) are numbers storing the specific transformation, which happened in the layer. The goal here is to find the right values of these weights (parameters) to get a correct value for target prediction. However, the DL model may contain millions of parameters, and each change in weight value may affect the behavior of the others. To add control over the output, a loss function is added. The loss function computes a distance of a predicted and a true target, and thus captures the accuracy of the model. A lower loss score, which refers to better accuracy, can be achieved by adding a feedback signal to adjust the values of the weights, and it is done by an optimizer. The initial layer weights are assigned randomly and thus the loss score is high. But by repeating this process (training loop/epoch), the weights ideally adjust in the correct direction and the loss score decreases [20].

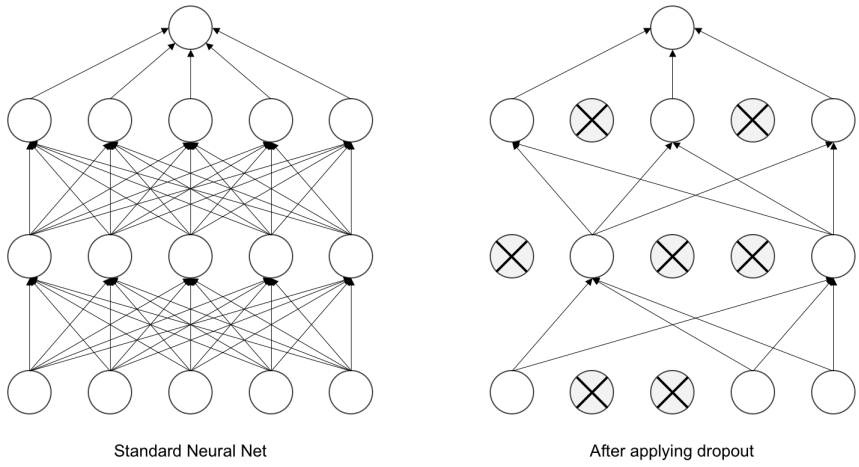


Fig. 3.2: Neural network vs. neural network with dropout[1]

When training a model, there is a risk of overfitting or underfitting. Overfitting can happen when the dimension of the chosen model is too complex for the data. The trained model does not generalize well, as it is too much fitted on the training data. Underfitting on the other hand means, that the model had the potential to be further trained to achieve higher accuracy without overfitting. Deciding the training length of the model can be tricky, but a comparison of training and validation loss might help. By a rule of thumb, the training and validation losses should be the same, eventually, the validation loss can be a bit higher. If the validation loss is significantly higher than the training loss, the model is overfitting and vice versa. The loss value for classification tasks is computed with categorical cross-entropy.

[1]Source: `https://www.oreilly.com/library/view/deep-learning-for/9781788295628/assets/d4d20bd7-192c-48e7-9da2-6d3ddc7929e7.png`

This function measures the distance between a probability distribution output of a network and the true label, the lower the distance, the higher the accuracy.

We can add a dropout layer as a form of regularization for the network to prevent overfitting. The dropout layer randomly turns a fraction of features of a layer to zero during the training. The idea behind it is to break up insignificant patterns in the output values, which would be otherwise memorized by the network. This process is pictured in Fig. 3.2.

To allow the network to learn complex patter in the data, we add activation functions into the network. The ability to add a non-linearity is important since most data used in DL are not simply linear. Additionally, the activation function can restrict the size of the layer's values. ReLu (rectified linear unit) is a popular activation function and is defined as $R(z) = max(0, z)$. It has many variations such as Leaky ReLu, SeLu, ELU, etc... Another activation function to mention is the softmax function, which is placed at the output layer in classification models. It normalizes the outputs and converts them into probabilities, which add up to one.

As for the optimizers, SGD and Adam are the two most common ones. Stochastic gradient descend (SGD) is based on gradient descend, which is descending along the slope of a function and is trying to find its minimum. While gradient descend is working with all the data, SGD randomly picks data points for the calculation to speed the process, it is however still slow. It can also use momentum, which accelerates the gradient descend and helps against staying at the local minimum. Adam optimizer combines the momentum concept from SGD and adaptive learning from Ada delta optimizer. The memory requirement for Adam is low, it is also computationally efficient, straightforward to implement and it is much faster than SGD.

3.2 Convolutional Neural Networks

Convolutional networks (CNNs) are designed for working with data with grid-like topology. They can be described as neural networks, that use convolution instead of general matrix multiplication in at least one of their layers [18]. Images data can be seen as a 2D grid of pixels and CNN architectures are currently state-of-the-art in most computer vision tasks. Time-series data have a 1D grid structure and can therefore be used in convolutional networks as well.

Convolution in general is an operation on two functions of a real argument, that describes how the shape of one function modifies the other one and can be described as:

$$s(t) = x(t) * w(t) = \int_{-\infty}^{+\infty} x(\tau)w(t-\tau)\,d\tau, \tag{3.1}$$

where x(t) is referred to as input, w(t) as kernel, and s(t) can be referred to as feature map.

Since the data stored on a computer are discrete, the integral can be replaced with a sum:

$$s(t) = x(t) * w(t) = \sum_{-\infty}^{+\infty} x(\tau)w(t-\tau). \tag{3.2}$$

The input data, as well as the kernel in machine learning (ML), are usually multidimensional and the convolution is often used over multiple axes at a time. Many neural network libraries implement cross-correlation (where the kernel is not flipped) instead of a convolutional operation, as the flipping is non-essential in ML.

The following equation describes "convolution" (cross-correlation) on a 2D image I with a 2D kernel K:

$$S(i,j) = K(i) * I(j) = \sum_m \sum_n I(i+m, j+n)K(m,n). \tag{3.3}$$

CNN mostly consists of a combination of three basic layers, a convolutional layer, a pooling layer, and a fully connected layer. The convolutional layer is not connected to every input data point and can only see a few neighboring data points at a time. How big the area is, is given by the size of the convolutional kernel, which is a weight matrix optimized by the algorithm. This allows the network to see both low-level and high-level features. Extracted features are then activated by an activation function, a commonly used one is a non-linear ReLu function. The pooling layer reduces the size of the input data. The two most common approaches are maximum pooling, or average pooling, where they use either the maximum value of the window, or calculate the average value. This layer does not contain any parameters and helps with reducing the size of the final model. This can help against overfitting and reduce required memory for computation. The fully connected (dense) layer is used at the end of the network to map the features into labels. These layers have neurons fully connected to the following layer. The final layer is an N-dimensional vector, where N is the number of the target classes from the dataset. In a classification task, the last dense layer is commonly activated with a softmax function which assigns the best-predicted class to the input.

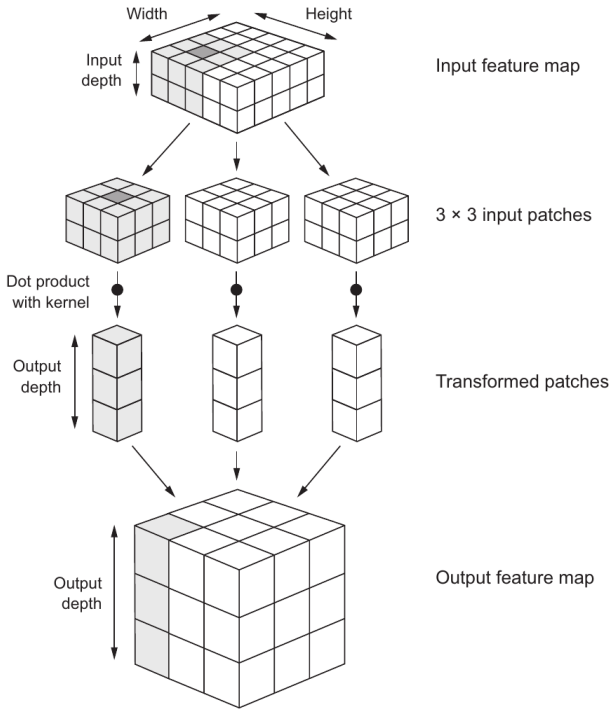


Fig. 3.3: Working principle of convolution [20]

3.3 Recurrent Neural Networks

While CNNs are good at processing data with spatial information, recurrent neural networks (RNNs) are designed to work with sequential information. They are well known for their application with text data in natural language processing but can work with any other sequential data such as audio signals, or data for forecasting stock prices, and much more...

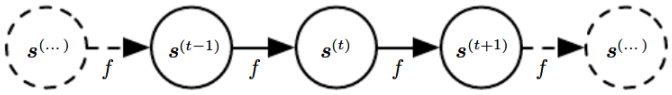


Fig. 3.4: Dynamical system illustrated as an unfolded computational graph [18]

In RNN, each member of the output is a function of the previous members of the output [18], as can be seen in Fig. 3.4. In other words, the networks iterate through the received sequence and maintain a state with relative information based on what it has already seen during this process. The network resets this state when a new sequence is passed to the network. A simple recurrent layer has a problem with vanishing gradient, which makes it harder to learn dependency during long

time steps. Long-short-term memory layer (LSTM) and gated recurrent unit layer (GRU) help to save this problem. Including an extra dropout layer, helps with the prevention of overfitting the network.

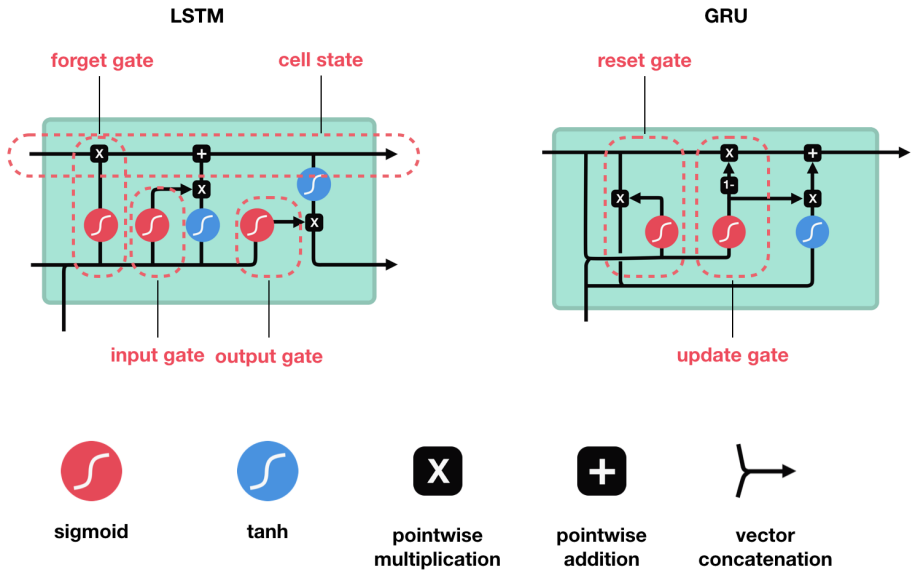


Fig. 3.5: LSTM and GRU Cell[2]

LSTM and GRU layers work similarly, and both save the information for later during the training and prevent older signals from gradually vanishing. Both LSTM and GRU have self-loop gates, which are controlled by a hidden unit, which allows the gradient to flow for a long duration. These gates can be seen in Fig. 3.5.

In a forget gate in the LSTM cell, information from a current input and a previous hidden state is passed through a sigmoid function. The sigmoid function outputs a value in a range between 0 and 1. Numbers close to 0 mean to forget, whereas numbers close to 1 mean to keep the information.

An input gate updates the information in the cell. It passes the current input and the previous hidden state through a sigmoid and a tanh function and multiplies their outputs. The output of the sigmoid function close to 1 means the values are important and the values close to 0 are not. The tanh function helps with regulating the network and outputs values between -1 and 1.

The calculation of the cell state is done by pointwise multiplication of the previous cell state and the output of the forget gate. This is followed by pointwise addition with an output of the input cell so that the cell state can be updated with new values relevant to the neural network.

[2]Source: `https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21`

At last, the output gates decides about the new hidden state. It holds the information on previous inputs and is used for predictions. The values are given by passing the previous hidden state and current input into a sigmoid function and are multiplied with the new cell state passed through a tanh function. Both the new cell state and new hidden state are passed into a further cell.

GRU is a newer generation, with fewer tensor operations, and should be therefore little faster than the LSTM layer. It contains only two gates - an update gate and a reset gate. The update gate has similar behavior as the forget and input gates from LSTM. The reset gate decides how much of the past information should be forgotten.

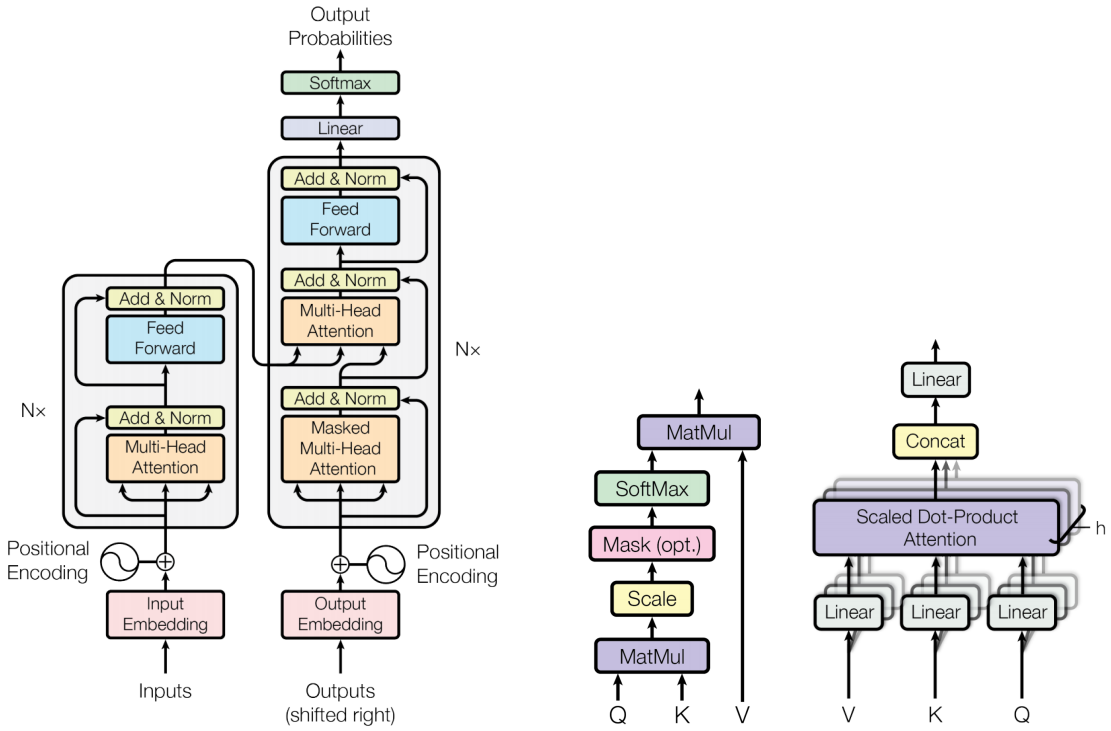
3.4 Transformers

A transformer model is based on a self-attention mechanism and works without any convolutional or recurrent layers. Simply said, attention layer looks at an input sequence and decides which parts of it are important - where to pay attention. Transformer architecture was introduced in [21] 2017 and was originally proposed for NLP applications. Transformers replaced the recurrent neural networks at NLP tasks and the focus shifts nowadays to CV tasks as well [22].

The architecture of the transformer, pictured in 3.6a consists of two main parts - an encoder (on the left) and a decoder (on the right). The encoder consists of two sub-layers, multi-head attention, and a fully connected feed-forward neural network, with additional residual connections. This layer can be repeated N-times before it is passed to the decoder. The decoder has the same sub-layers with additional multi-head attention over the encoder's output. This layer can be stacked N-times as well. This is followed by dense layers, where the last one is activated by a softmax function.

The input data is passed through an embedding layer, which for example would represent the text data in form of numerical vectors. The data is then positionally encoded so that the model will not lose information about the positions of each sequence element. Since the architecture does not contain any convolution or recurrence, positional encoding is needed to make use of the input sequence's order. This can be done by adding sine and cosine functions of different frequencies to the input data:

$$\begin{aligned} PE_(pos, 2i) &= \sin\left(pos/10,000^{2i/d_{model}}\right) \\ PE_(pos, 2i+1) &= \cos\left(pos/10,000^{2i/d_{model}}\right) \end{aligned} \tag{3.4}$$



(a) Transformer architecture (b) Scale Dot-Product Attention (c) Multihead Attention

Fig. 3.6: Overview of transformer architecture [21]

V, K, and Q in the attention functions are all vectors and stand for Value, Key, and Query. The function maps the query and a key-value pair to an output, which is a weighted sum of the values. The weights are computed by a compatibility function of the query and corresponding key [21]. This is pictured in Fig. 3.6b and can be described as:

$$Attention(Q, K, V) = softmax\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V. \tag{3.5}$$

The scale dot-product attention is applied multiple times via the multi-head attention, which is showed in 3.6c. This way each head has a chance to learn different representations of subspaces at different positions.

4 Proposed Architectures

This chapter introduces four proposed DL architectures for radio modulation classification. As mentioned earlier, the use of convolutional or recurrent neural networks is common for the task. What differs are the chosen hyper-parameters, such as number of layers, number of filters and kernels in the layers, connections between the layers, etc... This resulted in numerous different architectures proposed in other papers, with different complexity in their architectures and a different number of network parameters.

Two of the proposed architectures are inspired by a rather simple network design, which can be seen in [4,5]. The networks in those two papers contained over 2 million parameters. While a larger number of parameters means the network can learn more features from the data, it can also result in overfitting and a larger size of the final model. The architectures proposed in this thesis focus on a reduction of parameters for an easier deployment onto an embedded platform later on. While quantization of the model is an option to reduce the size of the model, post-training quantization resulted in an accuracy drop in [15]. Rather than using quantization to achieve a smaller size of the model, the architectures were hyper-tuned to result in a smaller number of parameters. The parameters were reduced up to 50 times compared to [4, 5], and the accuracy remained comparable.

The third design is a variation of the second one and includes a different type of recurrent layer. The last design is based on transformers [21]. They replaced recurrent networks in the NLP field, and have the potential to replace convolutional neural networks in the computer vision field as well [22]. With this promising background, it was intriguing to see, whether transformers could be applied to the modulation classification. Since I could not find any published papers, which would include an architecture with the transformer network, the implementation is based on my intuition.

The architectures were written in Python using a deep learning API Keras [20] utilizing the free GPU offered by Google Colab. The code for the following architectures is publicly available at a GitHub repository [23].

The first three architectures, CNN, CLDNN, and CGDNN use the Adam optimizer, with a starting learning rate between 0.002 and 0.0007 (depending on the network). A ReduceLROnPlateau callback from the Keras API was included to reduce the learning rate size if the validation loss does not improve for 3-5 epochs. The training is stopped with an EarlyStopping callback if the learning rate does not improve for 8 epochs, and a ModelCheckpoint callback then saves a model with the smallest validation loss. The MCTransformer uses an SGD optimizer and starts with a learning rate of 0.03, which can be lowered during the training.

4.1 CNN Architecture

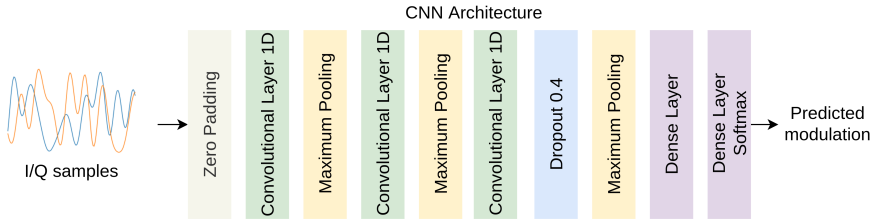
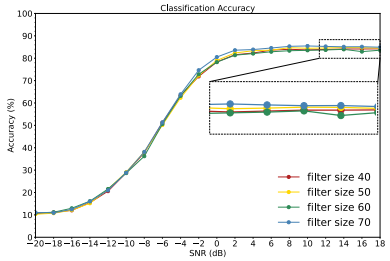
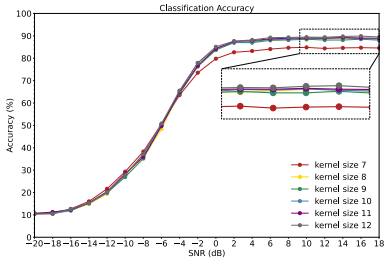


Fig. 4.1: CNN architecture [24]

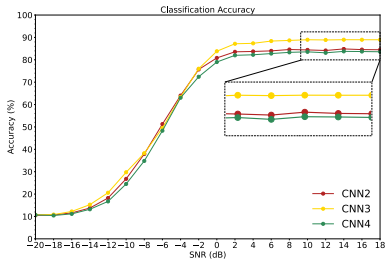
The first proposed architecture is a convolutional neural network, and it can be seen in Fig. 4.1. The design is rather simpler compared to other convolutional networks used in computer vision such as ResNet. However, it seems that the modulation classification is not such a complex task for the neural networks, as image classification is, and the networks can find enough features even with simpler designs.



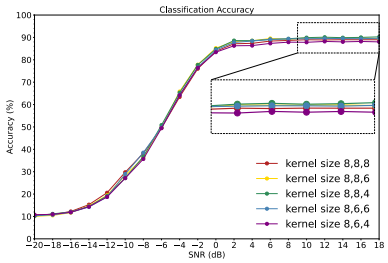
(a) Filter size of a conv layer



(b) Kernel size of a conv layer



(c) Varying numbers of conv layers



(d) Varying kernel sizes of conv layers

Fig. 4.2: Hyper-parameter tuning on CNN

As mentioned earlier, the design of this architecture was inspired by other papers such as [4] or [5], but the goal was to reduce the number of parameters. Fig. 4.2

shows the process of hyper-parameter tuning of the network. The performance difference between the filter and kernel sizes in Fig. 4.2a and Fig. 4.2b does not significantly differ for larger filter and kernel sizes. As those hyper-parameters can significantly affect the final number of the model's parameters, I opted for a filter size of 50 and kernel sizes of 8, 8, and 4, which proves to work well as it can be seen in Fig. 4.2d. Finally, Fig. 4.2c clearly shows, that three convolutional layers are the best choice for the task.

Tab. 4.1: Architecture of the CNN

Layer	Arguments	Output Shape
Input	Shape (128,2)	128x2
Zero padding 1D	Padding 4	136x50
Conv 1D	Filters 50, Kernels 8, ReLu	129x50
Max Pooling 1D	Pool size 2	64x50
Conv 1D	Filters 50, Kernels 8, ReLu	57x50
Max Pooling 1D	Pool size 2	28x50
Convolution	Filters 50, Kernels 4, ReLu	25x50
Dropout	Rate 0.6	25x50
Max Pooling 1D	Pool size 2	12x50
Flatten	-	600
Dense	Units 70, SeLu	70
Dense	Units N[1], Softmax	N

The final architecture consists mainly of three 1D convolutional layers. These layers produce a tensor of outputs by convolving the kernel over a single spatial dimension. The layers are activated by a rectified linear activation function (ReLu) to provide a non-linearity to the layers. The maximum pooling layers down-samples the length of the sequence, by taking a maximum value over a spatial window, which length is given by a chosen pooling size. A dropout layer is added to prevent potential overfitting and the network is finished by adding two fully connected layers. The last dense layer is activated by the Softmax activation function. The Softmax function is commonly used in classification tasks, as it will represent the output of the final layer as probabilities of each class, where all the values add up to 1. Table 4.1 offers an overview of the exact chosen parameters of the network. The overall number of parameters is 73,730[2].

[1]Size of N depends on the number of modulation classes in a chosen dataset

[2]All parameters mentioned in this chapter are for N=10, which is the number of classes for a RadioML2016.10b dataset. The final number may slightly differ, for different values of N.

4.2 CLDNN Architecture

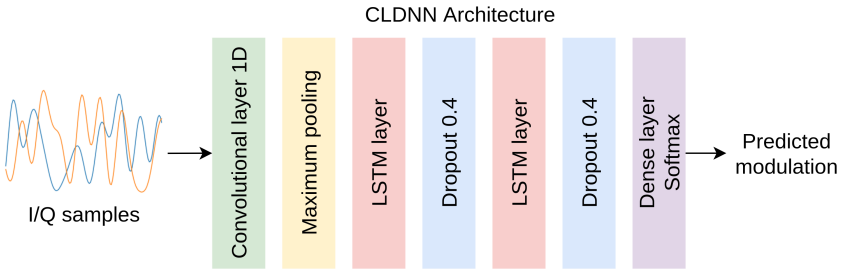


Fig. 4.3: CLDNN architecture [24]

The next proposed architecture is a convolutional long-short-term-memory deep neural network (CLDNN). The combination of convolutional and recurrent networks combines a partial feature extraction from the convolutional layer and a long-term memory coherence from the LSTM layers. The CLDNN architectures in [5] and [16] used three convolutional layers and one LSTM layer. However, the architecture proposed in this thesis is composed of a convolutional and a maximum pooling layer, followed by two LSTM layers with dropout layers, which should prevent overfitting, as I was able to reach better results this way. The final layer is once again a dense layer activated with a Softmax function. The overview of the CLDNN architecture can be seen in Fig. 4.3, and the arguments of the used layers are listed in Tab. 4.2. This architecture resulted in 105,546 parameters.

Tab. 4.2: Architecture of the CLDNN

Layer	Arguments	Output Shape
Input	Shape (128,2)	128x2
Conv 1D	Filters 64, Kernels 8, ReLu	121x64
Max Pooling 1D	Pool size 2	60x64
LSTM	Filters 64	60x64
Dropout	Rate 0.4	60x64
LSTM	Filters 64	60x64
Dropout	Rate 0.4	60x64
Flatten	-	3480
Dense	Units N, Softmax	N

4.3 CGDNN Architecture

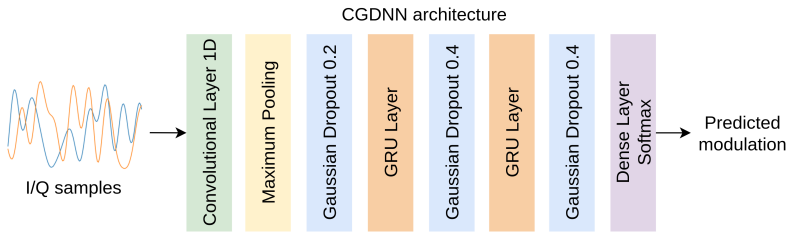


Fig. 4.4: CGDNN architecture

The third design is a convolutional gated recurrent deep neural network (CGDNN). The idea to use a GRU layer came from papers [25], where the authors use residual connections[3] between convolutional and GRU layers and paper [26], where the authors use a single GRU layer for the classification. Neither residual connections nor a single GRU layer, however, worked for me. As mentioned earlier, the GRU layer belongs to the recurrent networks group and its working principle is similar to LSTM. As was mentioned in chapter 3, the GRU has fewer hidden units, which requires less computation and makes the training faster. By switching the LSTM layers for GRU in the CLDNN architecture, while keeping the same hyper-parameters, the number of parameters was reduced by circa 15,000 and the computational time was reduced by one-fourth per epoch while maintaining the same accuracy. The parameters of this architecture were further reduced down to 49,690 with further hyper-parameter tuning, which can be overviewed in Tab. 4.3.

Tab. 4.3: Architecture of the CGDNN

Layer	Arguments	Output Shape
Input	Shape (128,2)	128x2
Conv 1D	Filters 80, Kernels 12, ReLu	117x80
Max Pooling 1D	Pool size 2	58x80
GRU	Filters 64	58x40
Gaussian Dropout	Rate 0.4	58x40
GRU	Filters 64	58x40
Gaussian Dropout	Rate 0.4	58x40
Flatten	-	2320
Dense	Units N, Softmax	N

[3]This is a technique which skips one or more layers in the networks.

4.4 MCTransformer Architecture

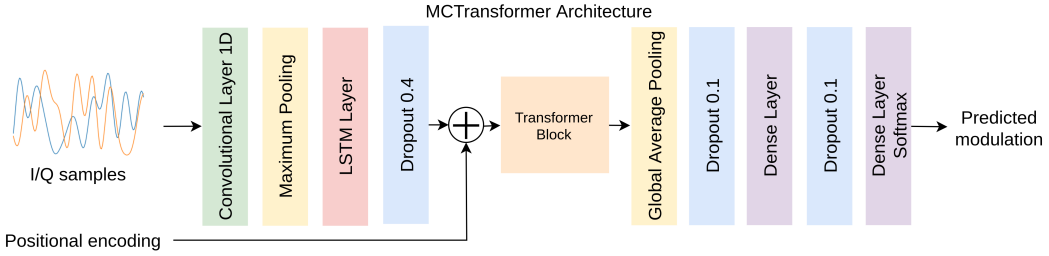


Fig. 4.5: MCTransformer architecture

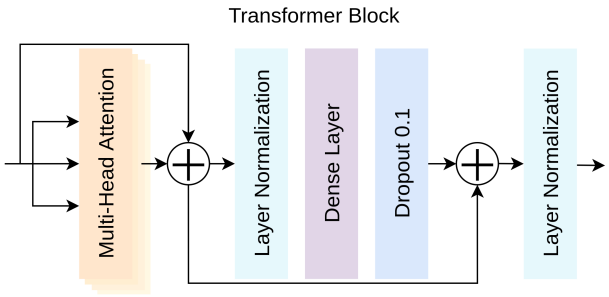


Fig. 4.6: Structure of a transformer block

The last architecture proposed in this thesis is made with transformers. I've mentioned in Chapter 3, that transformer networks do not contain any convolutional, nor recurrent layers, I have included them in this architecture, as can be seen in Fig. 4.5. The idea behind it is the following: When working with text data, the words are first encoded and represented as a vector of numbers. This means that if we work with a sentence that has 20 words and we encode each word to be represented as a vector of a length 128, the input in the transformer would be 20x128. The network is then able to adjust the values in the vectors in a way, where some similar words would be closer to each other and others would be further away, and the larger number of vectors allows more adjustments. However, the shape of the input data for the modulation classification is 128x2 for the data used in this thesis. My idea was to provide more input features to the transformer by passing it through the CNN and LSTM layers first. This way those two layers extract some features from the input data and the transformer can then decide, where to pay attention to them.

The transformer block is pictured in detail in Fig. 4.6. The attention mechanism has four heads and it is afterward passed through a normalization layer and a dense layer with a dropout. The output from the dense layer is concatenated with the output from the multi-head attention and is normalized once more before it is passed further on to the final few dense layers. Table 4.4 offers an overview of the architecture design, which had 104,374 parameters.

Tab. 4.4: Architecture of the MCTransformer

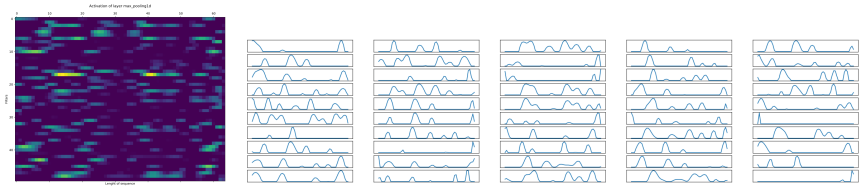
Layer	Arguments	Output Shape
Input	Shape (128,2)	128x2
Conv 1D	Filters 64, Kernels 8, ReLu	121x64
Max Pooling 1D	Pool size 2	60x64
LSTM	Units 64	60x64
Dropout	Rate 0.4	60x64
+ Positional Encoding	-	60x64
Transformer Block	Number of heads 4	60x64
Global Average Pooling 1D	-	64
Dropout	Rate 0.1	64
Dense	Units 20	20
Dropout	Rate 0.1	20
Dense	Units N, Softmax	N

4.5 Layer Visualization

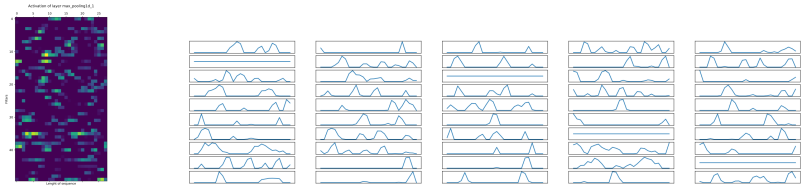
Deep learning models are often referred to as black boxes, as we pass data in, get out a solution to the problems, but the process in between is rather unclear to us. The data are passed through the layers and the network is trying to find out helpful features, which might not make sense to us. However, we can at least visualize what is happening with the data, as it gets passed through the trained network. This might not answer all the questions about the learning process in the network, but it can help with the idea behind it

The visualization shows activations in the layers and feature maps. As the data pass through the layers of the network, they activate certain parts of the filters or the gates. These activations are mapped onto the sequence which caused them creates a feature map, which provides us with the idea of what feature is relevant for a given filter.

1st Maximum Pooling layer



2nd Maximum Pooling layer



3rd Maximum Pooling layer

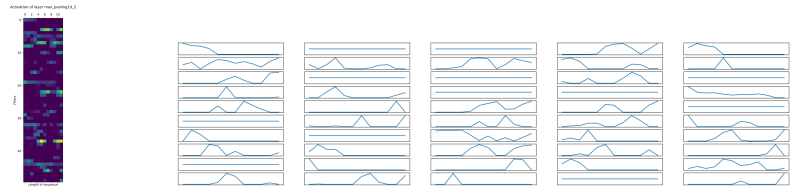


Fig. 4.7: Activation maps (left) and feature maps (right) of CNN layers

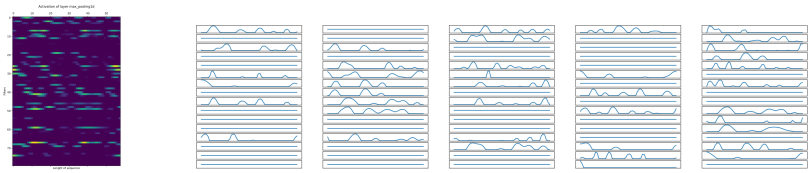
There are two figures included, one with the visualizations of the CNN model in 4.7 and the other of the CGDNN in 4.8. The color scheme for activation maps on the left side goes from dark blue to yellow and the brighter the color is, the more activated the given part is. While they do not give us much interesting feedback, as some images would[4], they offer a nice insight on the difference between convolutional and recurrent layers.

What we can see in Fig. 4.7 is a hierarchical structure of the CNN. Commonly, the first few layers in CNNs see more patterns and the deeper layers are more class-oriented. Again, this explanation would be rather clear with an image example, rather than a modulation signal. The convolutional layer can also only see a few neighboring data points at a time. This way it can learn local patterns and can apply this pattern at any position in the input sequence later on. This way of seeing the data in restricted windows is recognizable from the figures with the peaks and

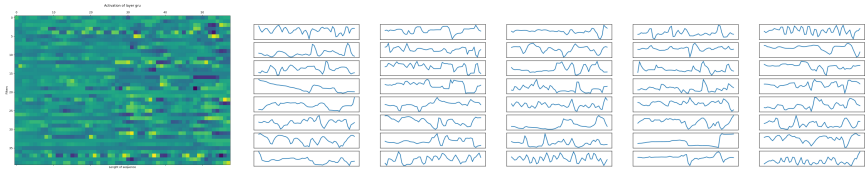
[4]With a picture of a dog, for instance, we would most probably see the main outlines in the first few layers and the deeper we would get, the more specific the features would get and some of the features could be a shape of an ear, or a nose, etc...

the stillness in between. This is even more noticeable if we compare these layers to the GRU layers in Fig. 4.8. Since the recurrent layers see the whole sequence all the time, while they decide which part carries relevant information, the activations are stronger over the whole sequence length.

Max Pooling layer



1st GRU layer



2nd GRU layer

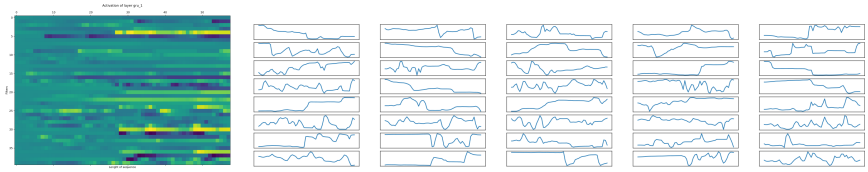


Fig. 4.8: Activation maps (left) and feature maps (right) of CGDNN layers

5 Dataset

When it comes to deep learning, the key to having a good final model for a specific task requires not only choosing a fitting architecture, but also a high-quality dataset. The performance of the model depends on the learned features learned from the input training data, which means, that if certain real-life conditions are ignored in the dataset, the trained model might not generalize well.

There are publicly available datasets for the modulation classification task, among which the RadioML datasets are most often mentioned in research papers. These datasets are either synthetical or measured in laboratory environments. The authors of the RadioML datasets believe, that since the steps in communication systems are often synthetic and deterministic, it is possible to use a synthetical dataset. In [27] the authors argue with a feature-based method and SVM classifier, that the classification results may deteriorate if real-life conditions are not captured in the training dataset.

While synthetical data or data measured in a laboratory environment might be good to get an idea of the network's possible performance, the need for a robust, real-life dataset is critical to obtain a robust modulation classifier. Modern communication systems are adaptive and have many configuration settings and many variables should be thought of when creating a representative robust dataset [28]. Since there is a lack of a public, robust dataset, the networks in this thesis were trained on synthetical and laboratory datasets, which are introduced in the following sections.

5.1 RadioML Datasets

In 2016 the DeepSig company published their first datasets determined for modulation classification [29]. Along with that, they published a paper [4], in which they demonstrated the use of convolutional neural networks for the AMC. Although this was not the first-ever published paper on this topic, it is very often cited as they allowed others easier access to the AMC task by publicly sharing the datasets. Thanks to that, the datasets are often used in other research papers, and it was possible to get a good results comparison for the proposed architectures.

Signals in the datasets are synthetical and were generated with GNU Radio software. To simulate real-life scenarios, channel model blocks, consisting of sample rate offset, center frequency offset, selective fading, and additive white Gaussian noise, were added. As a final step, the data was scaled to unity energy as a prior step to further usage of the dataset in the machine learning field.

The data are represented as 2x128 vectors of in-phase and quadrature signals (I/Q). They were generated for various signal-to-noise ratios (SNRs) in the range from -20 dB to 18 dB[1]. The versions RadioML2016.10a and 10b[2] used in this thesis contain digital and analog modulations such as BPSK, QPSK, 8PSK, QAM16, QAM64, CPFSK, GFSK, PAM4, AM-DSB, AM-SSB, and WBFM. The version 10a contains 220,000 samples and the version 10b has 1,200,000 samples.

5.2 Migou-Mod Dataset

The Migou-Mod Dataset [30] contains over-the-air measured signals. A transmitter used for their generation was formed by USRP B210 connected to a computer with GNU Radio software. This dataset includes the same 11 modulations as the RadioML dataset. The authors of this dataset also used the same source code and same data sources from the RadioML dataset. To record the signals, they used a MIGOU platform on the receiver side. The measurements for this dataset were carried out in an office environment at distances of 1 and 6 meters. The average corresponding SNRs are 37 dB and 22 dB respectively. The data are represented as 2x128 I/Q vectors and results in a total of 8.8 million samples.

5.3 VUT Dataset

This dataset is synthetical and was provided to me by my supervisor. It was generated in MATLAB with 1000 samples per SNR value and each modulation type. It includes three QAM modulation schemes and further OFDM, GFDM, and FBMC modulations which are not included in previous datasets. To mimic the RadioML dataset, the data are represented as 2x128 vectors of I/Q signals in the SNR range from -20 dB to 18 dB.

[1]However, note that the SNR values might not correspond with an actual SNR which is determined through spectral analysis. The issues with this dataset are analyzed and discussed in the following blogpost `https://cyclostationary.blog/2020/08/17/more-on-deepsigs-rml-data-sets/`. The values of the SNR seems to be changed by varying noise floor level. Some SNR values and modulations seem to have a very wide noise floor, which can result in wide varying SNR values. I have stuck to the convention of referring to those SNR parameters as "dB" values as it is done in the original papers of the authors of the dataset and all the papers which use it. But it should be kept in mind, that the actual values might be higher and diverse.

[2]The version 10b does not contain AM-SSB modulation

6 Results

This chapter presents results achieved on the proposed architectures and their discussion. The results are sectioned into three parts, one for each dataset, for easier comparison. They are presented in form of confusion matrices (CMs). The confusion matrix measures the performance of the classification problem, for N classes. The truth labels are placed along the y-axis and the predicted labels are placed along the x-axis. On the main diagonal of the CM correctly classified classes. Everything that lies outside the main diagonal was not correctly classified and it allows us to see commonly confused classes. The CMs below have a blue color scheme and the darker the shade of blue is, the more often was the classification prediction for the given class was made. Each cell in the matrix is annotated with the corresponding percentage for additional precision.

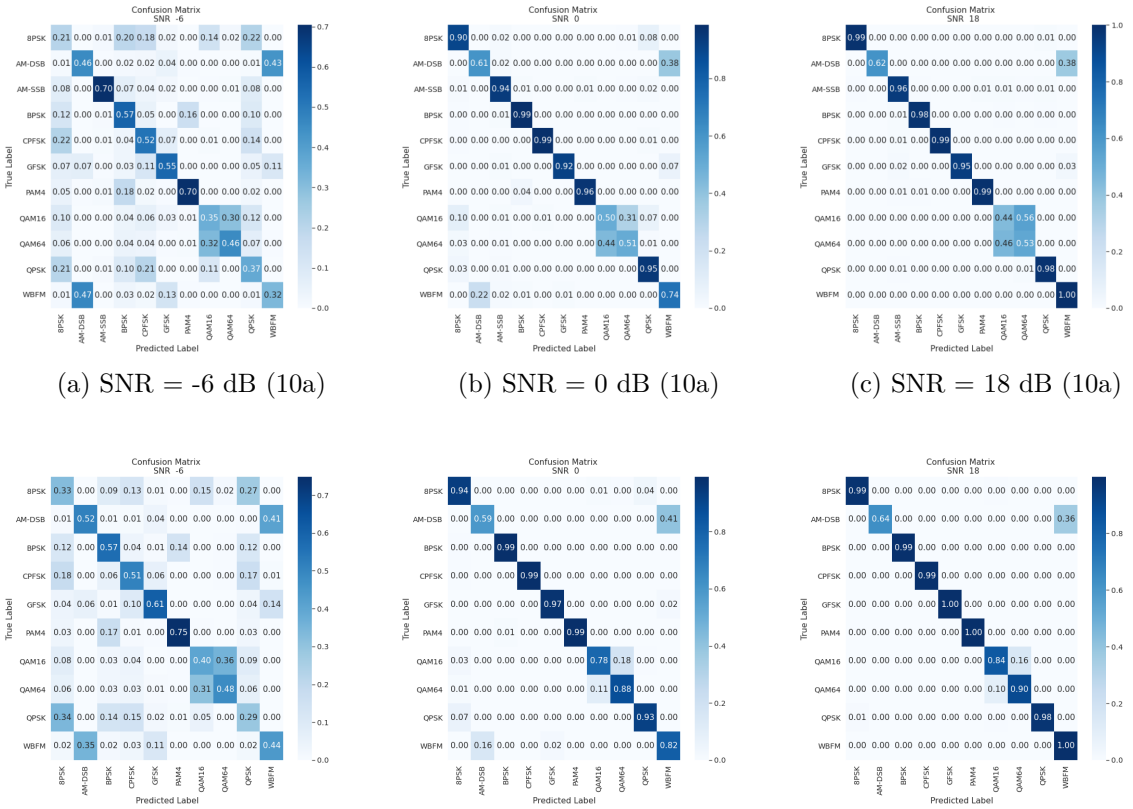
6.1 RadioML Datasets

The results on the RadioML2016.10a and 10b datasets are presented as first one. As a reminder, these datasets are very similar, and the only difference is in the size, where the 10a version consists of 220,000 signal samples (and an additional AM-SSB modulation), and the 10b version of 1,200,000. The only hyper-parameter which was changed when training the models on these two datasets was the batch size. The value was 128 for the 10a version and 256 for the 10b version, as this version is bigger, and increasing the number allowed faster training. Comparing the results of those two datasets together allows us to see the impact of the dataset size.

CNN

The results in Fig. 6.1 are of the CNN architecture. We can see, that the misclassification is quite strong in Fig. 6.1a and Fig. 6.1d, which represents lower signal-to-noise ratios. While the overall accuracy at the level -6 dB is only around 50%, we can see, that some of the classifiers were able to at least assign a correct modulation scheme - eg. the 8PSK gets confused for QPSK and BPSK, 16-QAM for 64-QAM, AM for FM and vice versa.

At SNR = 0 dB, we can see a huge improvement in the classification, where most of the modulation classes are classified correctly. There is a small confusion remaining between the QPSK and 8PSK, the classification of analog signals improved a little for both datasets. If we compare the QAM schemes classification for both datasets in Fig. 6.1b and Fig. 6.1e, we can see, how the size of the dataset might impact the classification. The model trained on the smaller dataset represented in



(a) SNR = -6 dB (10a) (b) SNR = 0 dB (10a) (c) SNR = 18 dB (10a)

(d) SNR = -6 dB (10b) (e) SNR = 0 dB (10b) (f) SNR = 18 dB (10b)

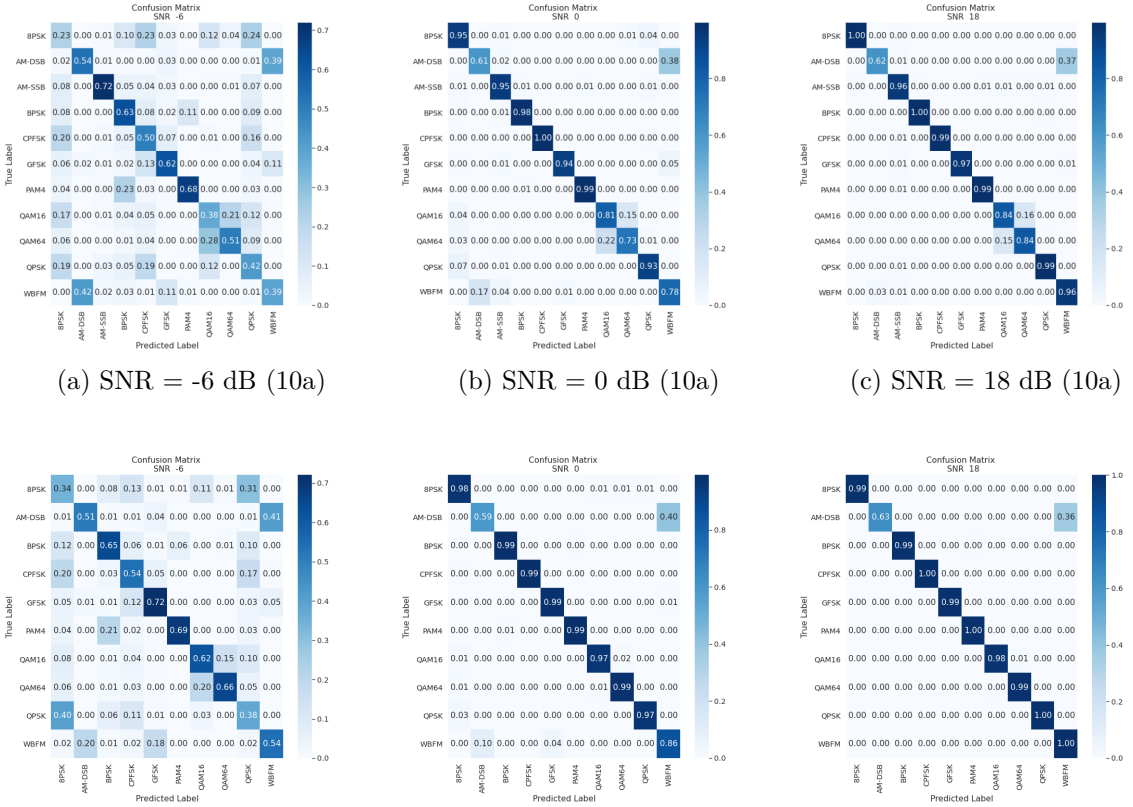
Fig. 6.1: Confusion matrices CNN - RadioML [31]

the 6.1b was able to classify the correct QAM class only for half of the samples. On the other hand, the model trained on the 10b version represented in 6.1e, which has 6x more samples per SNR and modulation class, was able to achieve an accuracy of around 80% for the QAM schemes.

By further increasing the SNR level, the only bigger noticeable change is with the confusion of WBFM for AM-DSB. The remaining confusion of AM-DSB for WBFM is common for this dataset. It is caused by data used for the generation of analog modulations. The authors of the dataset used voice records, which contained pauses in the speech. Since the length of the samples is rather short, some of them consist only of the pauses in the speech. The only available information in the signal is the carrier frequency, which is the same for all analog modulations, so the network learned to classify such occasions as WBFM modulation. The confusion between the QAM schemes remains unchanged even on the higher SNRs, and the network has difficulties in telling them apart. QAMs belong to high-order modulations as they can carry more bits of informations per symbol, which allows them to achieve faster data rates. But this comes at the price of low noise resistance. If we were

to look at constellation diagrams, we could see, that 64-QAM contains at the core same constellation points as the 16-QAM and since the signals can contain similar features, the classifier might find it harder to tell the QAMs apart. In this case, the CNN network did benefit from a larger dataset and more training data.

CLDNN



(a) SNR = -6 dB (10a) (b) SNR = 0 dB (10a) (c) SNR = 18 dB (10a)

(d) SNR = -6 dB (10b) (e) SNR = 0 dB (10b) (f) SNR = 18 dB (10b)

Fig. 6.2: Confusion matrices CLDNN - RadioML [31]

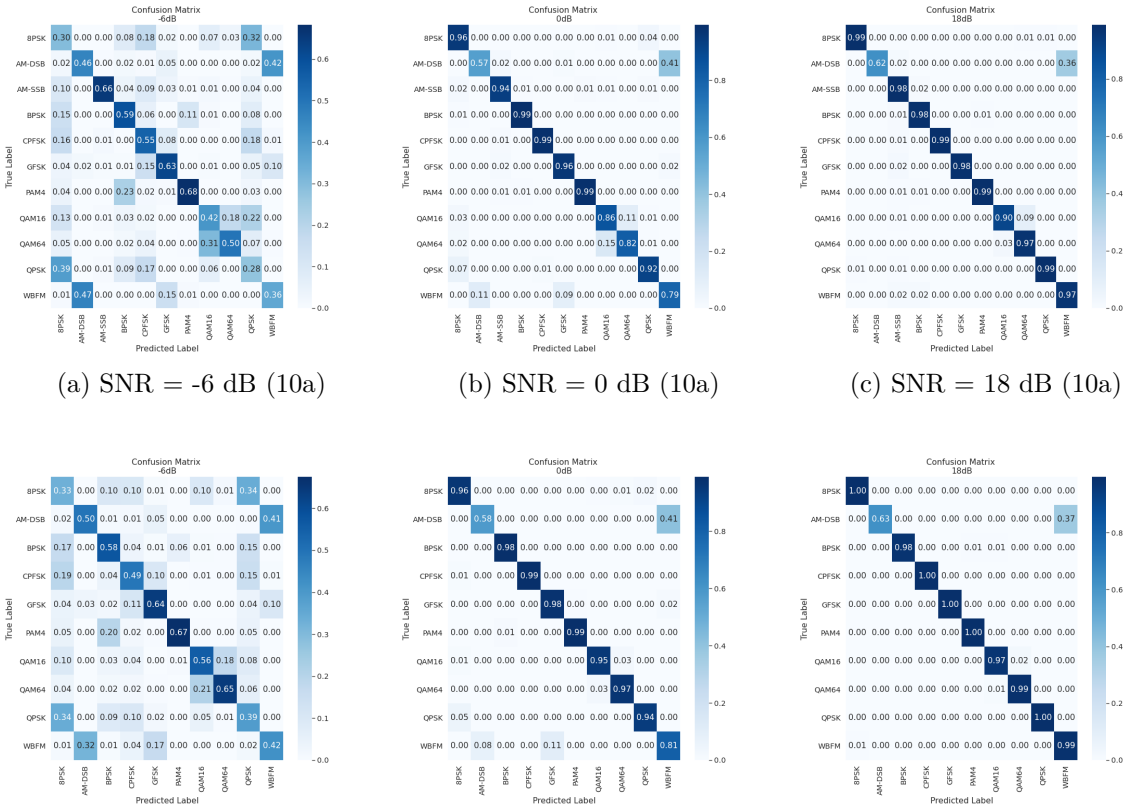
The confusion matrices of CLDNN architecture are represented in Fig. 6.2. At a first glance, the CMs at the SNR level -6 dB look very similar to the results from the CNN. However, if we were to look closer, we could notice an improved performance on the QAMs. This improvement is especially noticeable for the bigger 10b version of the two datasets in Fig. 6.2d, where the classification improved by 20% compared to the CNN result in Fig. 6.1d.

We can notice an improvement at the higher SNRs values as well. The classification accuracy of the QAMs for the 10a version improved up to 84%, which is an increment of 34% compared to the CNN. The CLDNN was able to improve the

accuracy of the WBFM by 10% and the QAMs reached accuracy above 97% on a version 10b at the SNR = 0 dB shown in Fig. 6.2e.

The combination of the convolutional and recurrent layers in this architecture seems to fit the problem better. The CLDNN architecture was able to find more suitable features for the different classes, especially the QAMs, compared to the CNN. While there is still a decrease in the accuracy for the smaller dataset, it is not as drastic anymore. The recurrent layers, however, slow the computing process down, and the training time per epoch was 2x longer, and bigger number of parameters also resulted in a bigger model size.

CGDNN



(a) SNR = -6 dB (10a) (b) SNR = 0 dB (10a) (c) SNR = 18 dB (10a)

(d) SNR = -6 dB (10b) (e) SNR = 0 dB (10b) (f) SNR = 18 dB (10b)

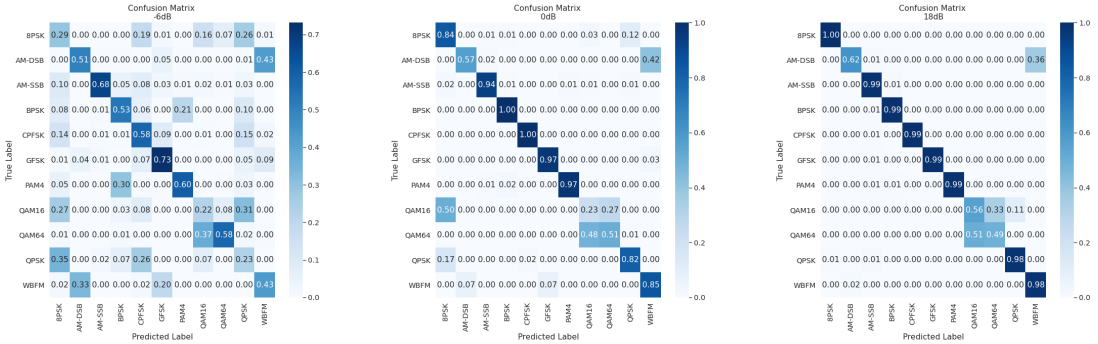
Fig. 6.3: Confusion matrices CGDNN - RadioML

The third set of confusion matrices belongs to the CGDNN model. This architecture consists of a combination of convolutional layers and recurrent layers as well. The overall number of parameters is the lowest out of all four proposed architectures, and the network is the smallest in size. This architecture is a good example of the

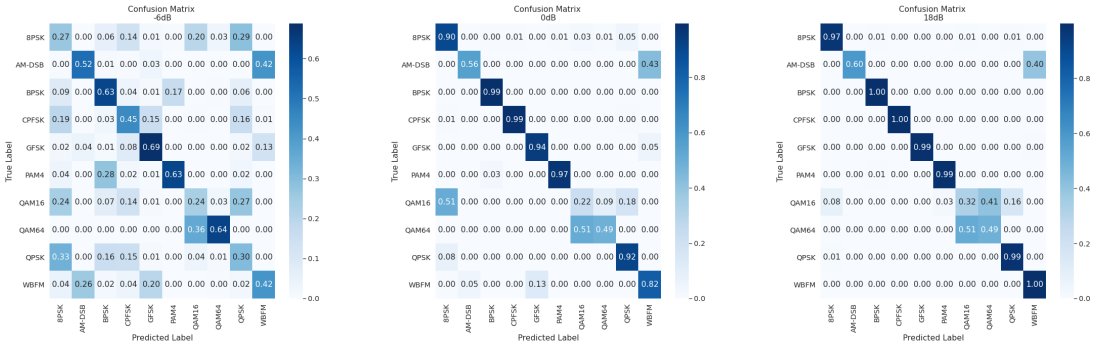
fact, that the models for radio modulation classification do not need to have a huge amount of parameters for better performance.

The confusion matrices in Fig. 6.3 show the best performance. The confusion matrices 6.3e and 6.3f for the larger 10b version show similar results as the CMs in Fig. 6.2, as there is only a tiny space left for an improvement. The improved performance can be therefore be seen on the smaller dataset in 6.3b on the QAM classes, as the accuracy improved by 5% at the SNR = 0 dB compared to the CLDNN. The confusion matrix in 6.3c shows an accuracy of 90% for the 16-QAM and 97% for the 64-QAM at the 18 dB SNR level.

MCTransformer



(a) SNR = -6 dB (10a) (b) SNR = 0 dB (10a) (c) SNR = 18 dB (10a)



(d) SNR = -6 dB (10b) (e) SNR = 0 dB (10b) (f) SNR = 18 dB (10b)

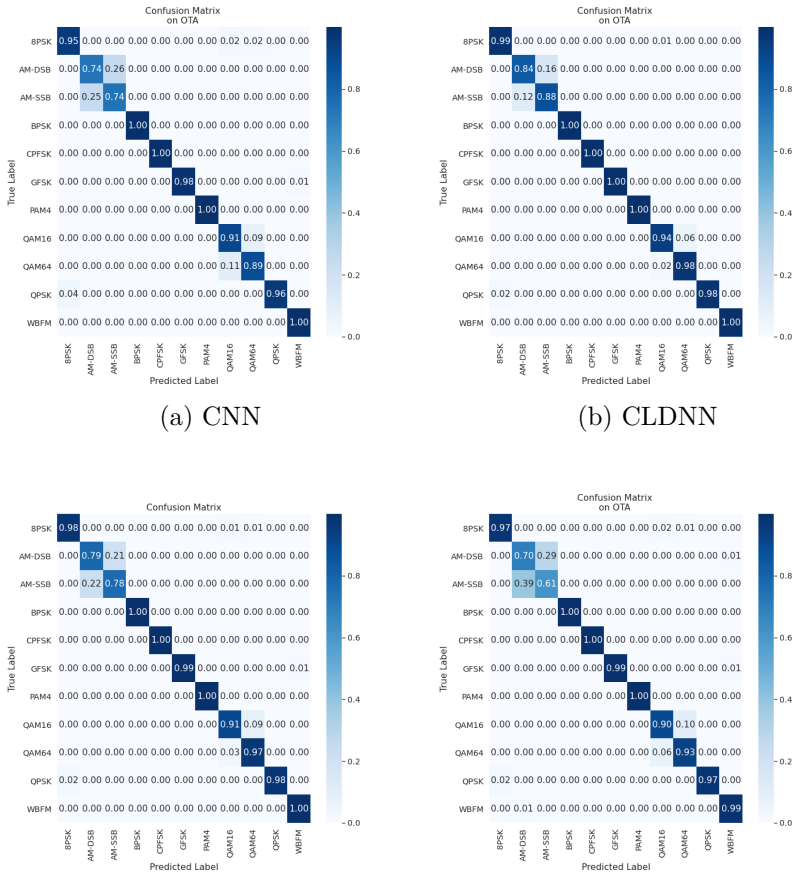
Fig. 6.4: Confusion matrices MCTransformer - RadioML

The last set of CMs for the RadioML datasets monitor the performance of the MCTransformer. As mentioned earlier, this architecture was included due to the success of the transformer architectures in NLP and newly in CV. However, it is noticeable from Fig. 6.4, that the MCTransformer achieved the worst results out of the four proposed architectures. The transformer model showed the lowest score out

of the four architectures on the 8PSK and QPSK, as well as on the QAM schemes at the SNR = 0 dB. And while the accuracy for the PSK schemes improved at higher SNR levels, the accuracy of the 16-QAM was at 32% and of the 64-QAM at 49%, as is to be seen in 6.4e and 6.4f.

The transformer had also problems with training on the unprocessed data extracted directly from the RadioML files. To train the model, the data needed to be normalized, by calculating a normal distribution. The original data which were in the range of -0.02 to 0.02 were afterward distributed between the values -3 and 3. The other architectures had no issue with the original distribution and worked well with both variations. While this solution might have a potential to be used for MC, it needs an improvement in its architecture, or at least the hyper-parameter tuning.

6.2 Migou-Mod Dataset



(a) CNN (b) CLDNN

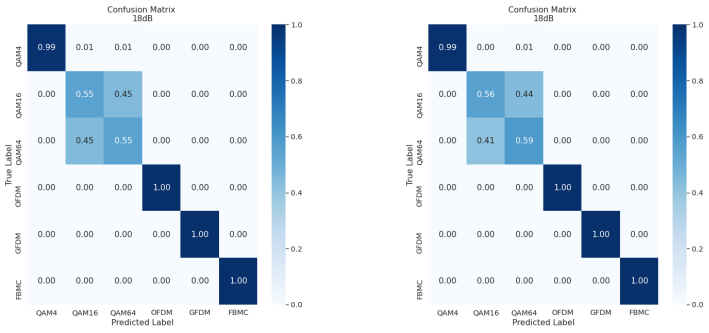
(c) CGDNN (d) MCTransformer

Fig. 6.5: Confusion matrices - Migou-Mod Dataset [24]

The Migou-Mod dataset consists of real-world signals measured in an office environment. The original file consists of 8.8 million signal samples divided into 11 classes and two SNR levels. The following results were achieved on a training set of the size 440,000, which holds both SNR levels. Since this dataset does not consist of a wide SNR range, there is a single confusion matrix for each DL model in Fig. 6.5. Since the SNR levels are relatively high enough, all four models were able to learn enough features and achieve high accuracy on this dataset. Since the dataset was created with the same data sources as the RadioML, the confusion between the analog classes is present in here as well. All four architectures also achieved over 90% accuracy on the QAM schemes. The CLDNN showed the best results, followed directly by the CGDNN.

6.3 VUT Dataset

The third section shows the confusion matrices of the VUT dataset. This dataset consists of three QAM schemes and of OFDM, GFDM, and FBMC modulations which are used in more current communication systems. The CNN and MCTransformer models have once again difficulties in telling 16-QAM and 64-QAM apart even at the highest SNR level of this dataset as can be seen in 6.6.



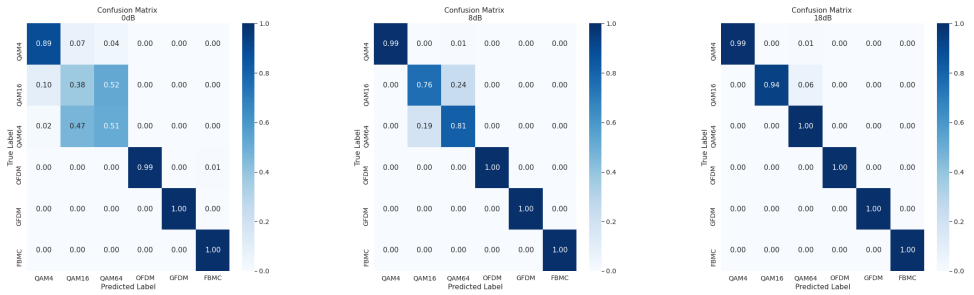
(a) CNN at SNR = 18 dB

(b) MCTransformer at SNR = 18 dB

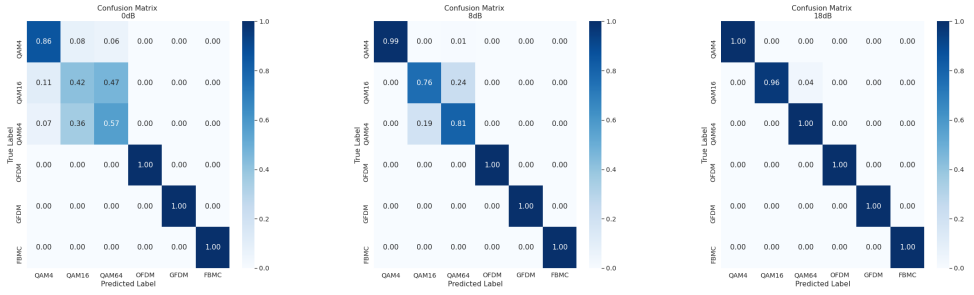
Fig. 6.6: Confusion matrices CNN and MCTransformer - VUT Dataset

On the other hand both CLDNN and CGDNN with their matrices in 6.7 showed good results on the QAM classification. The OFDM, GFDM, and FBMC modulations were successfully classified at SNR = 0 dB. The confusion matrices at this SNR level looked fairly similar for all DL models. We can see in on the CMs in 6.7b and 6.7e, that the accuracy for 16-QAM and 64-QAM is above 74% and gets above 94% at SNR = 18 dB in 6.7c and 6.7f. These are quite impressive results, if we consider,

that the 16QAM usually needs an SNR level of 18 dB and the 64 QAM at 24 dB at least to be properly classified. The overall accuracy of the models across the SNR range can be seen in 6.8.



(a) SNR = 0 dB (CLDNN) (b) SNR = 8 dB (CLDNN) (c) SNR = 18 dB (CLDNN)



(d) SNR = 0 dB (CGDNN) (e) SNR = 8 dB (CDGNN) (f) SNR = 18 dB (CGDNN)

Fig. 6.7: Confusion matrices CLDNN and CGDNN - VUT Dataset

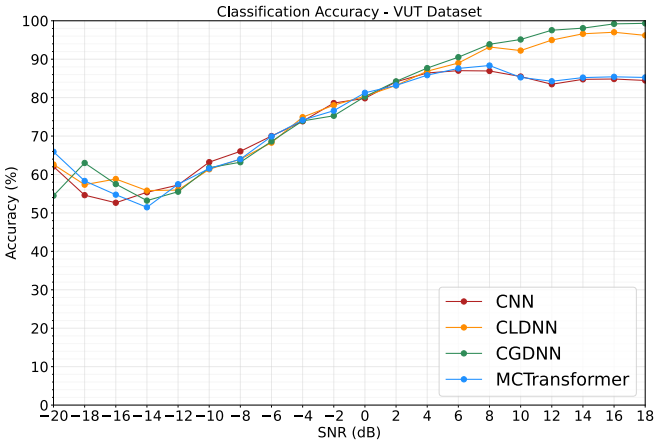


Fig. 6.8: Accuracy overview - VUT Dataset

6.4 Results Discussion

Judging by the results, represented so far in this chapter, the CLDNN and CGDNN are best suited for the radio modulation classification. The other two architectures, CNN and MCTransformer, proposed in this thesis were able to classify most of the modulations types but did struggle with the QAM schemes.

While the CGDNN model has the least parameters (≈50,000), it was the least affected by the reduction of the dataset size, which was represented with the versions of RadioML datasets in Fig. 6.3. Both the CLDNN and the CGDNN then reached very similar results on the Migou-Mod and VUT datasets, which were included for better evaluation of the architectures' performance. Since the performance of both networks is fairly similar, but the resulting size of the CGDNN is 636kB compared to 1.3MB of the CLDNN, I consider the CGDNN to be the best option of the four proposed models for the modulation classification.

It was mentioned earlier, that the RadioML datasets are open-source and often used by other researchers. The results from other papers were a good guideline for me for designing the architectures. The rest of this section is therefore dedicated to comparing the results from this thesis to the results achieved in other papers.

RadioML2016.10a

The Fig. 6.9 displays an accuracy overview on RadioML2016.10a over the SNR range. The accuracy achieved at SNR = 18 dB was 92.35% for the CLDNN and 92.18% for the CGDNN.

The first research paper [4] is written by the creators of the RadioML datasets. It introduces a CNN model and compares the results with other ML classifiers, such as Support Vector Machines, k-nearest neighbors, or random forests, which are used in the FB approach. The results of the CLDNN and CGDNN from this thesis have comparable and more stable results at SNRs above 0 dB. However, I was not able to reproduce the results from the paper [4] on SNRs below 0 dB, which outperformed my models by 10%. If we look the number of parameters of the models we would find out, that the CNN model [4] has over 2.8 million parameters [1], which is a 56 times higher number compared to the CGDNN model.

[1] `https://github.com/radioML/examples/blob/master/modulation_recognition/RML2016.10a_VTCNN2_example.ipynb`

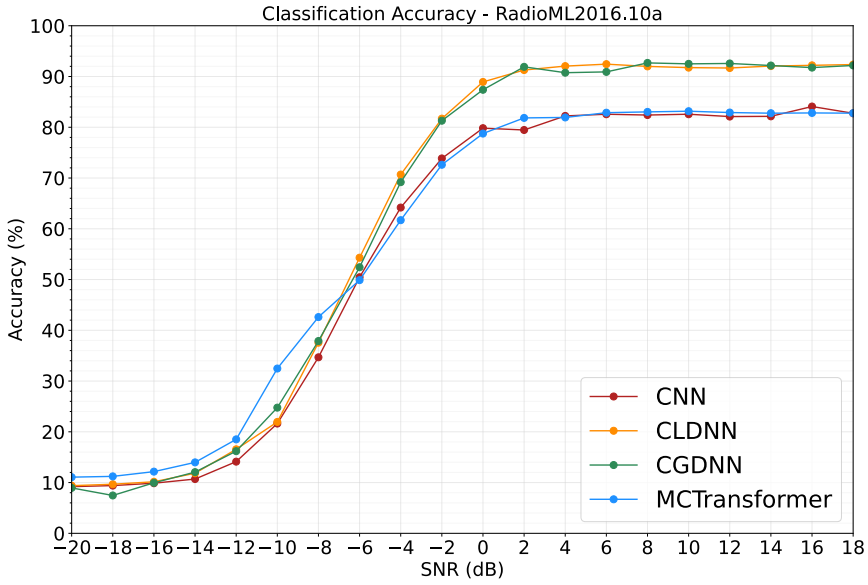


Fig. 6.9: Accuracy overview - RadioML2016.10a

The second research paper [15] to mention has two deep learning architectures - LSTM and CNN. The LSTM model from the paper, which performed better than the CNN model, was trained on the amplitude and phase data, rather than on the original in-phase and quadrature data. It achieved an average accuracy of 90% at high SNRs in the range of 0dB to 20dB. Although the proposed CLDNN and CGDNN models have lower accuracy by ≈5% compared to the LSTM model, they managed to outperform the LSTM by more than 2% above the SRN = 0 dB. The number of parameters for the LSTM was over 2 million and with that 40 times more compared to CGDNN.

The last paper [25] proposed a CGDNet model, which has three convolutional layers with residual connection, followed by a single GRU layer and three dense layers. This model has similar results as the CNN from paper [4], which outperform the networks proposed in this thesis. The CGDNet model also outperformed both of mine models at higher SNRs by 1%. The authors of this paper were able to reduce the number of parameters to 126,676, which is quite comparable to the CLDNN, but still to times higher if we look at the CGDNN.

Even though the performance of the CLDNN and CGDNN models was lower for the low SNRs, the models proposed in this thesis prove, that the DL models can perform well enough even with a lower parameter number.

RadioML2016.10b

The comparison of version 10b is included next. The overview of the accuracies can be seen in Fig. 6.10. The CLDNN reached 93.64% accuracy by the 18 dB SNR and the CGDNN was just below that number with 93.38%.

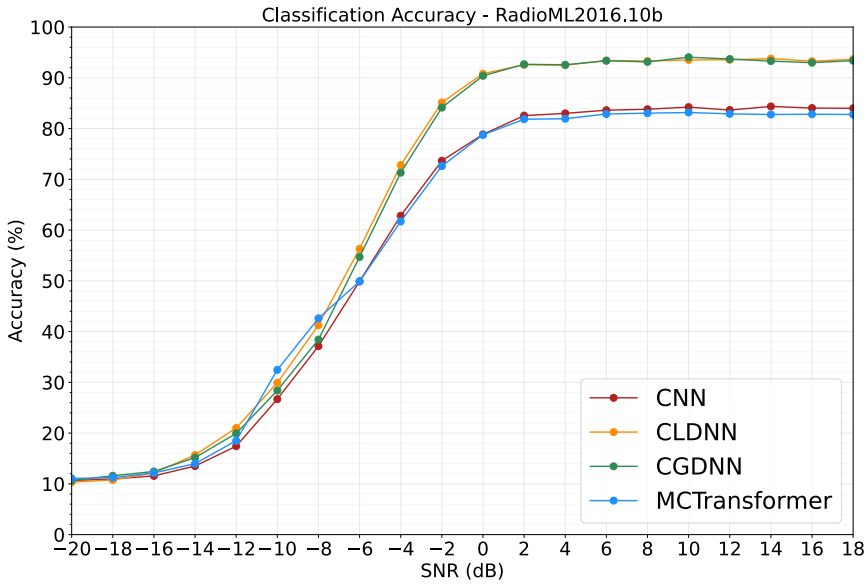


Fig. 6.10: Accuracy overview - RadioML2016.10b

In [5] the author focused on comparing CNN, LSTM, and CLDNN models. The results of this paper were similar to mine, and the CLDNN model from the paper showed the best performance with an accuracy of 91.8% at SNR 18 dB. The CLDNN and CGDNN models from this thesis were able to outperform the CLDNN model from the paper by more than 2%. Similarly to [15], the models in this paper contained over 2 million parameters.

The proposed CLDNN and CGDNN were able to outperform the results in [16] by more than 1% as well. The authors chose ResNet architecture and were able to achieve 92% accuracy at high SNRs.

This time the CLDNN and CGDNN were able to outperform the CGDNet model from the already mentioned paper [25] by more than 3% at high SNRs. Whether the authors could replicate the higher accuracy at the lower SNR levels was not specified for this dataset version.

Migou-Mod Dataset

The last comparison is done on the Migou-Mod dataset and the results, which the authors of the dataset achieved in [26]. The CLDNN and CGDNN models from this thesis were trained on a training set size of 440k and achieved accuracy was 96.35% and 94.35% respectively. The accuracy reached in the paper [26] with the closest comparable size of the training set with 550k was 92.3%. The best result mentioned in the paper was achieved with the signals measured at 1m (higher SNR) and a training set size of 94.6%. The authors used only a single GRU layer followed directly by a dense layer. Their memory footprint with 128 cells in the GRU layer was 207kB and is three times smaller than the memory footprint of the CGDNN. But when I was trying to evaluate this architecture with a single GRU layer on Radio ML datasets, the results were not as good. This shows, that while a single GRU layer may work on easy datasets, which have high SNRs, it might not be robust enough to replicate the results with noisy signals.

Conclusion

With the new 5G technology and the starting research in 6G, there is a focus on deep learning applications in communication systems. This thesis is focused on designing radio modulation classifiers with deep learning methods. There are currently two approaches to radio modulation classification - likelihood-based approach and feature-based approach. For this approaches, there is a big trade-off between the computational complexity, performance, and the number of modulation classes. The classifier also needs either prior knowledge about the signal or the need to extract features from the signal. Deep learning classifiers can be viewed as a third approach, which has been researched mainly over the past five years. The DL classifiers can work directly with the received data without any previous knowledge or feature extractions. They can be also trained to classify a large number of modulation types without any significant increase in the computational complexity. They can be easily upgraded to hold additional modulation classes or signals from another environment.

The output of this thesis are four deep learning architectures, which are publicly available in a GitHub repository [23]. The structure of the architectures, along with the achieved results on different datasets can be found in the chapters of this thesis. The main focus during the designing of the architectures was on parameter reduction of the DL models. Other papers very often present DL models with a large number of parameters, which affects the resulting model size. While there are techniques such as quantization, which can reduce the size of the model post-training, it can negatively affect the accuracy of the model.

The architectures proposed in this thesis demonstrate, that a properly hyper-tuned architecture can achieve better results even with a low number of parameters. To prove this, the results of the architectures from this thesis were compared to results shared in other research papers. The proposed CLDNN and CGDNN architectures were able to reach comparable results or outperform most other architectures. They have also reduced parameters up to 20 and 50 times respectively, compared to architectures from papers [4] and [5].

While diverse channel impairments can be simulated in the synthetical signals, real-world data measured in a non-laboratory set-up is needed. The lack of such a representative dataset is the biggest drawback of this work, as it can be crucial for the robustness of the classifier in real-life application. Due to the lack of a diverse real-world dataset at the moment, the models were not deployed on an embedded platform to be tested in real-life conditions. The scripts are, however, left at a state, where they are ready to be trained on new datasets. I would suggest using the CGDNN for further work, as the overall size of the trained model is 636kB, whereas

the CLDNN needs 1.3MB for the same accuracy.
Overall, I dare to say, that the deep learning approach shows potential to be used for radio modulation classification. The DL classifiers should be trained on robust datasets. Transfer learning[2] can make this approach easily re-usable and accessible for others. It will be surely interesting to watch, what the new research focused on DL application in communication systems will bring.

[2]Transfer learning uses weights of a pre-trained model as a starting point and can train on a much smaller dataset much faster and with higher accuracy than it would if it would train the model from scratch.

Bibliography

[1] DOWNEY, J., HILBURN, B.; O'SHEA, T. J; WEST, N. *Machine learning remakes radio* 2020, IEEE Spectrum, 57(5), pp. 35-39.

[2] ELSAYED, M.; EROL-KANTARCI, M. *AI-enabled future wireless networks: Challenges, opportunities, and open issues* IEEE Vehicular Technology Magazine, 2019, 14.3: 70-77.

[3] ERPEK, T.; O'SHEA, T. J.; SAGDUYU, Y. E.; SHI, Y.; CLANCY, T. C. *Deep learning for wireless communications. In Development and Analysis of Deep Learning Architectures* Springer, Cham., 2020, pp. 223-266

[4] O'SHEA T. J.; CORGAN J.; CLANCY T. C., Convolutional Radio Modulation Recognition Networks, In *International Conference on Engineering Applications of Neural Networks*, 2016, Communications in Computer and Information Science, vol 629, pp. 213–226, Springer, DOI: <https://doi.org/10.1007/978-3-319-44188-7_16>

[5] EMAM, A.; SHALABY, M.; ABOELAZM, A. M.; BAKR, A. E. H.; Mansour, A. A. H., A Comparative Study between CNN, LSTM, and CLDNN Models in The Context of Radio Modulation Classification, In *2020 12th International Conference on Electrical Engineering (ICEENG)*, Cairo, Egypt, 2020, pp. 190–195, DOI: <https://doi.org/10.1109/ICEENG45378.2020.9171706>

[6] PROAKIS, J. G.; SALEHIWALTER, M. *Digital communications* 5th ed. Boston: McGraw-Hill, 2008. ISBN 978–0–07–295716–7

[7] HAYKIN, S. S. *Digital communication systems* Hoboken, N.J.: Wiley, 2014. ISBN 978-0-471-64735-5

[8] XIONG, F. *Digital modulation techniques* 2nd ed. Boston: Artech House, 2006. ISBN 1-58053-863-0

[9] MARŠÁLEK, R. *Teorie rádiové komunikace* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2012. ISBN 978-80-214-4503-1

[10] FARUQUE,S. *Radio Frequency Modulation Made Easy* NY: Sprinter, 2017. ISBN 978-3-319-41200-9

[11] ZHU, Z.; NANDI, A. K. *Automatic modulation classification: principles, algorithms, and applications* Hoboken, N.J.: Wiley, 2014. ISBN 978-1-118-90649-1

[12] DOBRE, O.; ABDI, A.; BAR-NESS, Y.; SU, W. *A Survey of Automatic Modulation Classification Techniques: Classical Approaches and New Trends* Communications 2007, IET. 1. 137 - 156., DOI: 10.1049/iet-com:20050176.

[13] AL-NUAIMI, D. H.; HASHIM, I. A.; ZAINAL ABIDIN, I. S.; SALMAN, L. B.; MAT ISA, N. A. *Performance of Feature-Based Techniques for Automatic Digital Modulation Recognition and Classification—A Review* Electronics 2019, 8, 1407. https://doi.org/10.3390/electronics8121407

[14] WEST N. E.; O'SHEA T. J., Deep architectures for modulation recognition, In *2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, March 2017, pp. 1–6, DOI:<https://doi.org/10.1109/DySPAN.2017.7920754>

[15] RAJENDRAN, S.; MEERT W.; GIUSTINIANO, D.; LENDERS V.; POLLIN, S., Deep Learning Models for Wireless Signal Classification With Distributed Low-Cost Spectrum Sensors, In *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, September 2018, DOI: <https://doi.org/10.1109/ICEENG45378.2020.9171706>

[16] RAMJEE, S.; JU, S.; YANG, D.; LIU, X.; GAMAL, E. A.; ELDAR, C. Y., Fast Deep Learning for Automatic Modulation Classification, January 2019 Available: <https://arxiv.org/abs/1901.05850>

[17] FIGUEIREDO, D.; FURTADO, A.; OLIVEIRAY, R., Modulation Classification using Joint Time and Frequency-domain Data, In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, Antwerp, Belgium, 2020, pp. 1–5, DOI: <https://doi.org/10.1109/VTC2020-Spring48590.2020.9128493>

[18] GOODFELLOW, I.; BENGIO Y.; COURVILLE, A. *Deep learning* Cambridge, MA: MIT Press, 2016. Adaptive computation and machine learning series. ISBN 9780262035613, <http://www.deeplearningbook.org>

[19] ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. *Dive into Deep Learning*, 2020, <https://d2l.ai>

[20] CHOLLET, F. *Deep learning with Python* Shelter Island, NY: Manning, 2018. ISBN 9781617294433

[21] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT J.; JONES, L.; GOMEZ, A. N.; Kaiser, L.; POLOSUKHIN, I. *Attention Is All You Need*, 2017, Available: <https://arxiv.org/abs/1706.03762>

[22] DOSOVITSKIY, A., et al.; *An image is worth 16x16 words: Transformers for image recognition at scale*, arXiv preprint arXiv:2010.11929, 2020.

[23] PIJACKOVA, K., Radio Modulation Recognition Networks, GitHub, 2021 [Online] Available: <https://github.com/KristynaPijackova/Radio-Modulation-Recognition-Networks>

[24] PIJACKOVA, K., *Evaluation of CNN and CLDNN architectures on Radio Modulation Datasets*, In Proceedings of the 27th Conference STUDENT EEICT 2021, 4 pages, ISBN: 978-80-214-5942-7

[25] NJOKU, J. N.; MOROCHO-CAYAMCELA, M. E.; LIM, W.; *CGDNet: Efficient Hybrid Deep Learning Model for Robust Automatic Modulation Recognition*, IEEE Networking Letters, 2021.

[26] UTRILLA, R.; et al.; *Gated recurrent unit neural networks for automatic modulation classification with resource-constrained end-devices*, IEEE Access, 2020, vol. 8, pp. 112783-112794.

[27] DE VRIEZE, C.; SIMIC L.; MAHONEN P.; *The importance of being earnest: Performance of modulation classification for real RF signals*, 2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN) 2018 Oct 22 (pp. 1-5). IEEE.

[28] HALL,T. A.; CAROMI, R.; SOURYAL, M.; WUNDERLICH; A.; *Reference Datasets for Training and Evaluating RF Signal Detection and Classification Models*, 2019 IEEE Globecom Workshops (GC Wkshps), 2019, pp. 1-5, DOI: 10.1109/GCWkshps45667.2019.9024532.

[29] O'SHEA, T. J.; WEST, N., Radio Machine Learning Dataset Generation with GNU Radio, In *Proceedings of the GNU Radio Conference*, vol. 1, n. 1, September 2016, Available: <https://pubs.gnuradio.org/index.php/grcon/article/view/11>; Dataset available at: <https://www.deepsig.ai/datasets>

[30] UTRILLA, R., *MIGOU-MOD: A dataset of modulated radio signals acquired with MIGOU, a low-power IoT experimental platform*, Mendeley Data, V1, 2020, DOI: <http://dx.doi.org/10.17632/fkwr8mzndr.1>

[31] PIJACKOVA, K.; GOTTHANS, T., *Radio Modulation Classification Using Deep Learning Architectures*, 2021 31st International Conference Radioelektronika (RADIOELEKTRONIKA), 2021, pp. 1-5, DOI: 10.1109/RADIOELEKTRONIKA52220.2021.9420195.

Symbols and abbreviations

AMC	Automatic Modulation Classification
AWGN	Additive white Gaussian noise
CLDNN	Convolutional Gated Recurrent Deep Neural Network
CLDNN	Convolutional Long-Short-Term-Memory Deep Neural Network
CM	Confusion Matrix
CNN	Convolutional Neural Network
CR	Cognitive Radio
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
DSA	Dynamic Spectrum Access
FB	Feature-based
GRU	Gated Recurrent Unit
I/Q	In-Phase/Quadrature
LB	Likelihood-based
LSTM	Long-Short-Term-Memory
MC	Modulation Classification
ML	Machine Learning
NLP	Natural Language Processing
RNN	Recurrent Neural Network
SDR	Software Defined Radio
SoA	State-of-the-Art
SNR	Signal-to-Noise Ratio